PPCI7443

PCIﾊﾞｽ ４軸 ｻｰﾎﾞﾓｰﾀ/ｽﾃｯﾋﾟﾝｸﾞﾓｰﾀ用
軸制御基板

ユーザーズガイド

(Version : 2.00)

NPM 日本パルスモーター株式会社

目次

序論 1

1.1 特徴 4
1.2 仕様 5
1.3 ｿﾌﾄｳｪｱ支持 6
1.3.1 ﾌﾟﾛｸﾞﾗﾐﾝｸﾞﾗｲﾌﾞﾗﾘ 6
1.3.2 「PPCI7443 Utility」 6

インストール 7

2.1 付属品 7
2.2 PPCI7443 の外形図 8
2.3 ﾊｰﾄﾞｳｪｱｲﾝｽﾄｰﾙ 9
2.3.1 ﾊｰﾄﾞｳｪｱ構成 9
2.3.2 PCI ｽﾛｯﾄ選択 9
2.3.3 ｲﾝｽﾄｰﾙ手順 9
2.3.4 ﾄﾗﾌﾞﾙｼｭｰﾃｨﾝｸﾞ 9
2.4 ｿﾌﾄｳｪｱﾄﾞﾗｲﾊﾞｲﾝｽﾄｰﾙ 10
2.5 CN1 ﾋﾟﾝ番号： 外部電源入力 10
2.6 CN2 ﾋﾟﾝ番号： ﾒｲﾝｺﾈｸﾀ 11
2.7 CN3 ﾋﾟﾝ番号： ﾊﾟﾙｻｰ入力 12
2.8 CN4 ﾋﾟﾝ番号： 同時 START/STOP 12
2.9 CN5 ﾋﾟﾝ番号： 汎用 TTL 出力 13
2.10 ｼﾞｬﾝﾊﾟｰの設定 13
2.11 ﾃﾞｨｯﾌﾟｽｲｯﾁの設定 14

信号接続 15

3.1 ﾊﾟﾙｽ出力信号 OUT, DIR 16
3.2 ｴﾝｺｰﾀﾞﾌｨｰﾄﾞﾊﾞｯｸ信号 EA, EB, EZ 18
3.3 原点信号 ORG 20
3.4 ｴﾝﾄﾞﾘﾐｯﾄ信号 +EL, -EL 21
3.5 ｽﾛｰﾀﾞｳﾝ信号と PCS 信号 SD/PCS 22
3.6 ｲﾝﾎﾟｼﾞｼｮﾝ信号 INP 23
3.7 ｱﾗｰﾑ信号 ALM 24
3.8 偏差ｶｳﾝﾀｸﾘｱ信号 ERC 25

3.9 ｻｰﾎﾞｵﾝ信号 SVON 26
3.10 ﾚﾃﾞｨ信号 RDY 27
3.11 比較出力信号 CMP 28
3.12 位置ﾗｯﾁ入力信号 LTC 29
3.13 ﾊﾟﾙｻｰ入力信号 PA, PB 30
3.14 同時 START/STOP 信号 STA, STP 31
3.15 汎用の TTL 出力信号 ED 32

操作原理 33

4.1 動作制御モード 33
4.1.1 パルス出力モード 34
4.1.2 一定速度の動作 37
4.1.3 台形動作 38
4.1.4 Ｓ字曲線動作 40
4.1.5 ２～４軸の直線補間 42
4.1.6 ２軸円弧補間 46
4.1.7 加速/減速機能付き円弧補間 48
4.1.8 速度と加速時間の関係 49
4.1.10 原点復帰モード 57
4.1.11 手動ﾊﾟﾙｻｰﾓｰﾄﾞ 65
4.1.12 ﾀｲﾏｰﾓｰﾄﾞ 65
4.1.13 ﾊﾟﾙｻｰによる補間 66
4.2 ﾓｰﾀﾄﾞﾗｲﾊﾞｲﾝﾀｰﾌｪｰｽ 67
4.2.1 ＩＮＰ 67
4.2.2 ＡＬＭ 68
4.2.3 ＥＲＣ 68
4.2.4 ＳＶＯＮ，ＲＤＹ 69
4.3 ﾘﾐｯﾄｽｲｯﾁｲﾝﾀｰﾌｪｰｽと入出力状態 70
4.3.1 ＳＤ/ＰＣＳ 70
4.3.2 ＥＬ 71
4.3.3 ＯＲＧ 72
4.4 カウンタ信号 73
4.4.1 ｺﾏﾝﾄﾞｶｳﾝﾀ 73
4.4.2 ﾌｨｰﾄﾞﾊﾞｯｸｶｳﾝﾀ 73
4.4.3 ﾎﾟｼﾞｼｮﾝｴﾗｰｶｳﾝﾀ 75
4.4.4 汎用ｶｳﾝﾀ 75
4.4.5 ﾀｰｹﾞｯﾄﾎﾟｼﾞｼｮﾝﾚｺｰﾀﾞ 77
4.5 複数の PPCI7443 の操作 78

4.6 ｵｰﾊﾞｰﾗｲﾄﾞ機能（速度、位置） 79
4.6.1 速度のｵｰﾊﾞｰﾗｲﾄﾞ 79
4.6.2 位置のｵｰﾊﾞｰﾗｲﾄﾞ 83
4.7 位置の比較とラッチ 86
4.7.1 PPCI7443 のｺﾝﾊﾟﾚｰﾀ 86
4.7.2 位置の比較 87
4.7.3 位置のラッチ 90
4.8 ﾊﾞｯｸﾗｯｼｭ補正と振動抑制 91
4.9 ｿﾌﾄﾘﾐｯﾄ機能 92
4.10 割り込み信号制御 93
4.11 ｱｲﾄﾞﾘﾝｸﾞ制御 98

PPCI7443 Utility 99

5.1 PPCI7443 Utility の起動 100
5.2 PPCI7443 Utility について 100
5.3 PPCI7443 Utility の画面 101
5.3.1 メイン画面 101
5.3.2 入出力ｲﾝﾀｰﾌｪｰｽ構成画面 101
5.3.3 ﾊﾟﾙｽ I/O と割り込み構成画面 103
5.3.4 軸操作画面 104

ﾌｧﾝｸｼｮﾝﾗｲﾌﾞﾗﾘ 111

6.1 ﾌｧﾝｸｼｮﾝﾗｲﾌﾞﾗﾘのﾘｽﾄ 111
6.2 C/C++ﾌﾟﾛｸﾞﾗﾐﾝｸﾞﾗｲﾌﾞﾗﾘ 118
6.3 初期設定 119
6.4 ﾊﾟﾙｽ入力/出力構成 122
6.5 速度モード動き 124
6.6 １軸のﾌﾟﾘｾｯﾄﾓｰﾄﾞ動作 127
6.7 直線補間動作 132
6.8 円弧補間動作 139
6.9 原点復帰 145
6.10 パルサー動作 148
6.11 軸の動作状態 151
6.12 ｻｰﾎﾞﾄﾞﾗｲﾊﾞｲﾝﾀｰﾌｪｰｽ 152
6.13 I/O 状態の読み込み 155
6.14 割り込み信号制御 156

6.15 位置制御とｶｳﾝﾀ 162
6.16 位置の比較とﾗｯﾁ 165
6.17 連続動作 171
6.18 複数軸の同時操作 172
6.19 汎用ＴＴＬ出力 176

接続例 177

7.1 配線の概略 177
7.2 ｻｰﾎﾞﾄﾞﾗｲﾊﾞとの接続例 178

このガイドを使用する方法

このマニュアルは、ﾕｰｻﾞｰがPPCI7443を使用する時の手助けとなるように記述されています。このマニュアルはﾕｰｻﾞｰの要求に応えられる様にPPCI7443の様々な設定を変更する方法等を説明しており、それは全部で７章に分かれています。

第１章「序論」

基板の特徴、アプリケーション、及び仕様の概略を説明します。

第２章「インストール」

PPCI7443をインストールする方法を説明します。

第３章「信号接続」

PPCI7443のｺﾈｸﾀのﾋﾟﾝ番号と外部と接続する方法について説明します。

第４章「操作原理」

PPCI7443の詳細操作について説明します。

第５章「PPCI7443 Utility」

Windowsﾍﾞｰｽのﾕｰﾃｨﾘﾃｨﾌﾟﾛｸﾞﾗﾑ(PPCI7443 Utility)を使用してPPCI7443の初期設定とﾃｽﾄ動作の方法を説明します。

第６章「機能ライブラリ」

C/C++とVB言語でのﾌﾟﾛｸﾞﾗﾐﾝｸﾞ方法について説明します。

第７章「接続例」

PPCI7443とｻｰﾎﾞﾄﾞﾗｲﾊﾞの接続例を示します。

1

序論

PPCI7443 は PCI バス上で動作する４軸の軸制御基板です。本基板はｽﾃｯﾋﾟﾝｸﾞﾓｰﾀまたはｻｰﾎﾞﾓｰﾀを動作させるために高周波数のﾊﾟﾙｽを発生させることができます。動作機能では、任意の２軸の円弧補間、２～４軸の直線補間、そして連続した補間動作が行えます。また、ｵｰﾊﾞｰﾗｲﾄﾞ機能も位置と速度の両方が可能です。１台のシステムで複数の PPCI7443 基板を使用することができます(最高 12 枚)。４軸すべてにあるｲﾝｸﾘﾒﾝﾀﾙｴﾝｺｰﾀﾞにより、機械的な原因で発生する位置ズレが少なくなります。そして基板内の「FIFO」を使用して、正確で極めて速い位置比較を行うことができ、そして CPU の能力を費やすことなくﾄﾘｶﾞｰ出力します。さらに、機械的なｾﾝｻｲﾝﾀｰﾌｪｰｽ、ｻｰﾎﾞﾓｰﾀｲﾝﾀｰﾌｪｰｽ、および汎用の入出力信号が用意されています。

図 1.1 は PPCI7443 基板の機能ブロック図を示します。PPCI7443 は１個の PCL6045 を使用して４軸の軸制御を行っています。この基板の制御機能には直線、Ｓ字曲線での加速/減速、２軸の円弧補間と２～４軸の直線補間、位置決めにおける連続動作と 13 種類の原点復帰機能が PCL6045 によって行われます。複雑な計算を必要とするこれらの機能が内部的に PCL6045 に行われるので、ＣＰＵは他のタスクを自由に監視することができます。

「PPCI7443 Utility」は Windows ﾍﾞｰｽのﾕｰﾃｨﾘﾃｨｿﾌﾄで PPCI7443 を使用したｱﾌﾟﾘｹｰｼｮﾝ開発をｻﾎﾟｰﾄするのに備えられています。「PPCI7443 Utility」はﾌﾟﾛｼﾞｪｸﾄの初期段階の制御ｼｽﾃﾑをﾃﾞﾊﾞｯｸﾞするのにおいて非常に役立ちます。この画面には、PPCI7443 すべての軸の情報と入出力の状態を表示しています。「PPCI7443 Utility」に加えて、DOS と Windows 用に C++と Visual Basic 言語のﾌｧﾝｸｼｮﾝﾗｲﾌﾞﾗﾘが用意されています。また、ﾌｧﾝｸｼｮﾝﾗｲﾌﾞﾗﾘの使用方法の参考にｻﾝﾌﾟﾙﾌﾟﾛｸﾞﾗﾑが用意されています。

図 1. 2 はｱﾌﾟﾘｹｰｼｮﾝを開発するのに、このﾏﾆｭｱﾙを使用する過程を示すﾌﾛｰﾁｬｰﾄです。また、それぞれの章の詳細について対応する章をご参照下さい。

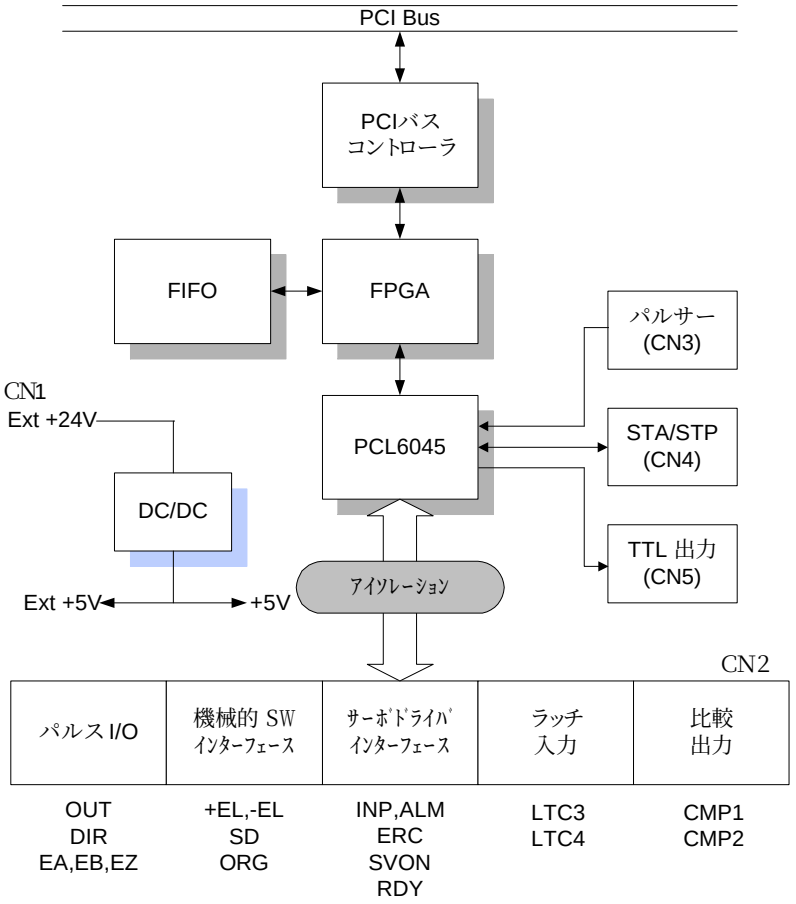


図 1.1 PPCI7443 のブロック図

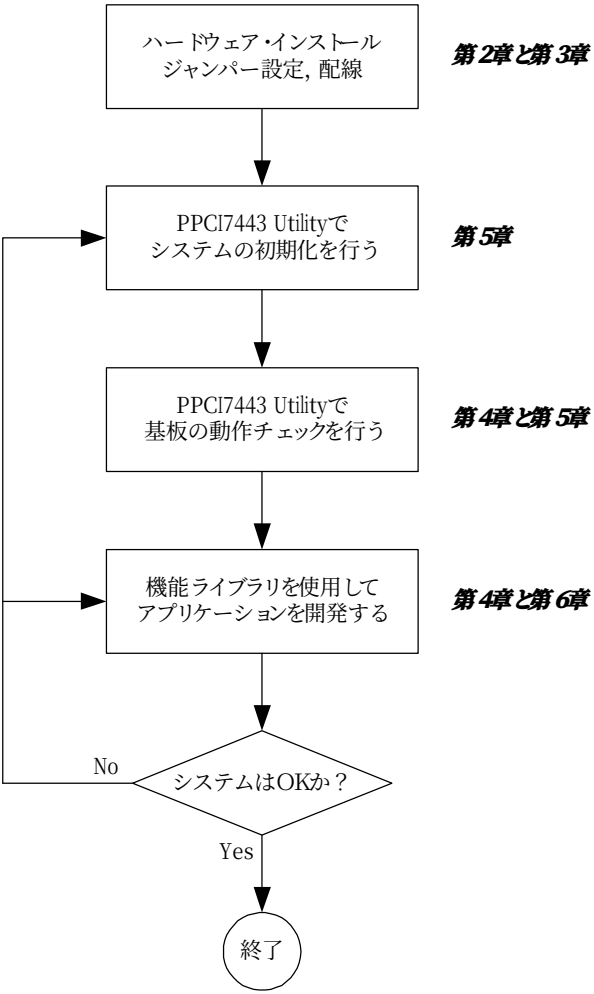


図 1.2 ｱﾌﾟﾘｹｰｼｮﾝを作成するﾌﾛｰﾁｬｰﾄ

1.1　特徴

下記のリストは PPCI7443 軸制御基板のの主な特徴をまとめています。

- 32ビットのPCI-バス、プラグ＆プレー
- ｽﾃｯﾋﾟﾝｸﾞﾓｰﾀもしくはｻｰﾎﾞﾓｰﾀ４軸を駆動（制御）
- 最高出力周波数は6.55Mpps
- ﾊﾟﾙｽ出力はOUT/DIRまたはCW/CCWの選択
- 加速時間、減速時間の設定ができる
- 台形動作、Ｓ字曲線動作が可能
- 任意の２軸円弧補間
- 任意の２～４軸の直線補間
- 連続補間動作
- 位置と速度のオーバーライド機能
- 比較機能による速度変化
- 13種類の原点復帰モード
- バックラッシュ補正機能と振動抑制機能
- ソフトウェアリミット機能
- ｲﾝｸﾘﾒﾝﾀﾙｴﾝｺｰﾀﾞは28ﾋﾞｯﾄのUP/DOWNｶｳﾝﾀを使用
- 原点ｾﾝｻｰ、EZ相信号、±EL信号が各軸に用意されている
- 後ろ２軸は位置データによる高速ラッチ機能。
- 4K(4096)のFIFOが自動に読み込まれ、前２軸は位置比較ﾄﾘｶﾞｰ信号を出力する
- 割り込み信号の制御
- 複数軸における同時ｽﾀｰﾄ/ｽﾄｯﾌﾟ動作
- 手動のパルサー入力
- １個のシステムで最大12枚(48軸)まで操作可能
- ｺﾝﾊﾟｸﾄなﾊｰﾌｻｲｽﾞPCB
- Windowsﾍﾞｰｽの「PPCI7443 Utility」の使用でｱﾌﾟﾘｹｰｼｮﾝ開発が容易
- PPCI7443ﾗｲﾌﾞﾗﾘとDOSﾗｲﾌﾞﾗﾘ用のﾕｰﾃｨﾘﾃｨとWindows 95/98/NT/2000/XP/Vista/7(32Bit) DLL

1.2　仕様

- 適応モータ：
  - ﾊﾟﾙｽ列入力のｽﾃｯﾋﾟﾝｸﾞﾓｰﾀﾄﾞﾗｲﾊﾞｰを使用したｽﾃｯﾋﾟﾝｸﾞﾓｰﾀ
  - ﾊﾟﾙｽ列入力のｻｰﾎﾞﾄﾞﾗｲﾊﾞを使用したACもしくはDCｻｰﾎﾞﾓｰﾀ
- 性能：
  - 制御軸数：　４軸
  - 最高出力周波数：　6.55Mpps（直線、台形かＳ字曲線速度動作による）
  - 基準クロック：　19.6608MHz
  - 位置パルス設定範囲：　-134,217,728 ～ +134,217,728ﾊﾟﾙｽ (28ﾋﾞｯﾄ)
  - UP/DOWMｶｳﾝﾀ範囲：　0 ～ 268,435,455(28ﾋﾞｯﾄ) or -134,217,728 ～ +134,217,727
  - ﾊﾟﾙｽﾚｰﾄ範囲設定(Pulse Ratio = 1 : 65535)

    0.1 pps ～ 6553.5 pps （乗数 = 0.1）
    1 pps ～ 65535 pps （乗数 = 1）
    100 pps ～ 6553500 pps （乗数 = 100）
- 入出力信号：
  - 各軸それぞれの入力/出力信号
  - パルス出力：　OUT, DIR
  - ｲﾝｸﾘﾒﾝﾀﾙｴﾝｺｰﾀﾞ信号：　EA, EB
  - ｴﾝｺｰﾀﾞ Ｚ相信号：　EZ
  - 機械的なスイッチ：　+EL, -EL, SD/PCS, ORG
  - ｻｰﾎﾞﾄﾞﾗｲﾊﾞ用I/F：　INP, ALM, ERC
  - 位置ラッチ入力：　LTC
  - 位置比較出力：　CMP
  - 汎用のデジタル出力：　SVON
  - 汎用のデジタル入力：　RDY
  - パルサー信号入力：　PA, PB
  - 同時のStart/Stop入出力：STA, STP
- 汎用出力
  - ６個のTTLﾃﾞｼﾞﾀﾙ出力：　ED0 ～ ED5

- ◆ 一般仕様
  - コネクタ： SCSI-ﾀｲﾌﾟの100ﾋﾟﾝｺﾈｸﾀ
  - 運転温度： 0℃ ～ 50℃
  - 保存温度： -20℃ ～ 80℃
  - 湿度： 5% ～ 85% 結露しないこと
- ◆ 消費電力:
  - スロット電源(入力)： +5V DC±5%, 900mA Max
  - 外部電源(入力)： +24V DC±5%, 500mA Max
  - 外部電源(出力)： +5V DC±5% 500mA Max
  - 寸法: 185 mm(L) X 98.4mm(H)

1.3 ｿﾌﾄｳｪｱ支持

1.3.1 ﾌﾟﾛｸﾞﾗﾐﾝｸﾞﾗｲﾌﾞﾗﾘ

プログラムを作成するﾕｰｻﾞｰのために、MS-DOS Borland C3.1用ﾗｲﾌﾞﾗﾘとと Windows 95/98/NT/2000/XP/Vista/7(32bit) 用に DLL と Visual Basic, Visual C++用ﾗｲﾌﾞﾗﾘを提供します。これらのﾌｧﾝｸｼｮﾝﾗｲﾌﾞﾗﾘは基板と一緒に出荷されます。

1.3.2 「PPCI7443 Utility」

詳細については第５章をご参照下さい。

2

インストール

本章は PPCI7443 をｲﾝｽﾄｰﾙする方法を説明します。下記の方法に従って、PPCI7443 をｲﾝｽﾄｰﾙして下さい。

- 付属品をﾁｪｯｸする。(ｾｸｼｮﾝ2.1)
- 基板をﾁｪｯｸする。(ｾｸｼｮﾝ2.2)
- ﾊｰﾄﾞｳｪｱをｲﾝｽﾄｰﾙする。(ｾｸｼｮﾝ2.3)
- ｿﾌﾄｳｪｱﾄﾞﾗｲﾊﾞをｲﾝｽﾄｰﾙする。(ｾｸｼｮﾝ2.4)
- 入出力信号接続(第3章)と操作原理(第4章)を理解する。
- ｺﾈｸﾀのﾋﾟﾝ番号(ｾｸｼｮﾝ2.5～2.11)を理解して配線する。

2.1 付属品

この取扱説明書に加えて、ﾊﾟｯｹｰｼﾞには下記の付属品があります。

- PPCI7443軸制御基板
- ｲﾝｽﾄｰﾙ CD
- DC+24V電源入力用ｹｰﾌﾞﾙ (CN1)
- 入出力信号用ｹｰﾌﾞﾙ (CN2)
- ﾊﾟﾙｻｰ入力用ｹｰﾌﾞﾙ (CN3)
- 同時ｽﾀｰﾄ/ｽﾄｯﾌﾟ用ｹｰﾌﾞﾙ (CN4)
- 汎用出力用ｹｰﾌﾞﾙ (CN5)

2.2　PPCI7443の外形図

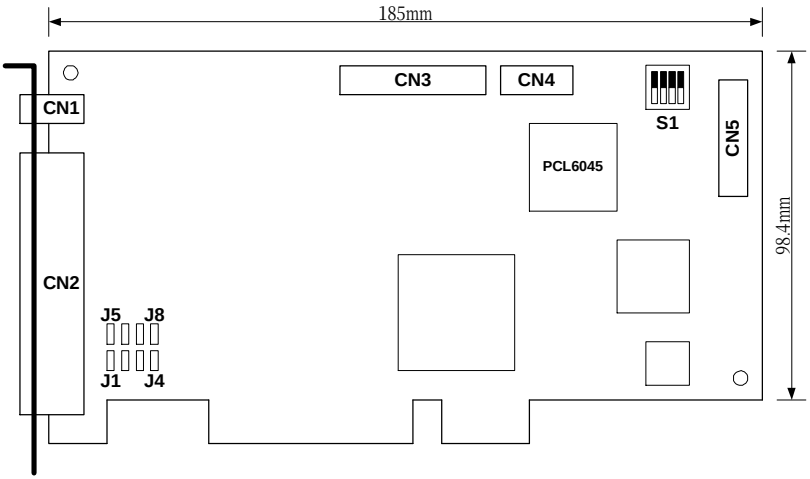
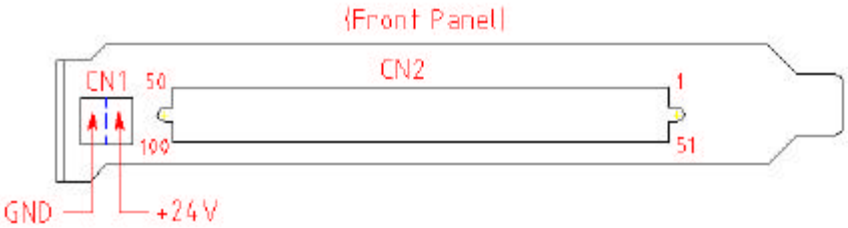


図 2.1 PPCI7443 の部品配置図

番号	使用目的	型式	ﾒｰｶｰ	備考
CN1	外部電源入力用ｺﾈｸﾀ	2317RJ-02	Neitron	付属
CN2	入力/出力信号ｺﾈｸﾀ	749197-1	AMP	付属
CN3	手動ﾊﾟﾙｻｰ信号入力ｺﾈｸﾀ	2317RJ-12	Neitron	付属
CN4	同時 Start/Stop 用ｺﾈｸﾀ	2317RJ-06	Neitron	付属
CN5	汎用出力用ｺﾈｸﾀ	CA2110SA100	CviLux	付属
S1	ｴﾝﾄﾞﾘﾐｯﾄの論理設定			
J1-J8	ﾊﾟﾙｽ出力ﾀｲﾌﾟの設定			



2.3 ﾊｰﾄﾞｳｪｱｲﾝｽﾄｰﾙ

2.3.1 ﾊｰﾄﾞｳｪｱ構成

PPCI7443 は、PCI ｺﾝﾄﾛｰﾗ用 IC を使用して、ﾌﾟﾗｸﾞ&ﾌﾟﾚｰで動作します。PCI ﾎﾞｰﾄﾞのﾒﾓﾘ使用法(入出力ﾎﾟｰﾄ位置)はｼｽﾃﾑの BIOS によって割り当てられます。ｱﾄﾞﾚｽの割付はｼｽﾃﾑのすべての PCI ﾎﾞｰﾄﾞごとのﾍﾞｰｽで行われます。

2.3.2 PCIｽﾛｯﾄ選択

ｺﾝﾋﾟｭｰﾀには、PCI と ISA ｽﾛｯﾄの両方があります。PC/AT ｽﾛｯﾄに PCI ｶｰﾄﾞを力ずくで挿入しないで下さい。どんな PCI ｽﾛｯﾄでも PPCI7443 を使用することができます。

2.3.3 ｲﾝｽﾄｰﾙ手順

1. ｱﾌﾟﾘｹｰｼｮﾝに合ったｼﾞｬﾝﾊﾟｰに設定して下さい。
2. ｺﾝﾋﾟｭｰﾀに接続されたすべてのｱｸｾｻﾘｰ(ﾌﾟﾘﾝﾀｰ、ﾓﾃﾞﾑ、ﾓﾆﾀｰなど)を外し、電源を OFF にして下さい。
   ｺﾝﾋﾟｭｰﾀのｶﾊﾞｰを取り外します。
3. 32 ビットの PCI 拡張ｽﾛｯﾄを選択して下さい。PCI のｽﾛｯﾄは ISA あるいは EISA ｽﾛｯﾄより短く、通常白いかもしくは象牙色です。
4. PPCI7443 を扱う前に身体に蓄積された静電気を放出して下さい。
   基板は、端を持って部品には触れないようにして下さい。
5. 基板をﾕｰｻﾞｰが選択した PCI ｽﾛｯﾄに挿入して下さい。
6. ｽﾛｯﾄから取り外したねじを使用して、ｼｽﾃﾑﾕﾆｯﾄの後部ﾊﾟﾈﾙに基板をしっかり固定して下さい。

2.3.4 ﾄﾗﾌﾞﾙｼｭｰﾃｨﾝｸﾞ

もし正しい場所に PCI ﾎﾞｰﾄﾞを挿入したにもかかわらず、ｼｽﾃﾑが起動しないか不安定な動作を起こした場合は割り込みの衝突によって起きていると思われます。対策としては、単純な見落としでないならばｼｽﾃﾑの BIOS 文献をご参照下さい。

2.4 ｿﾌﾄｳｪｱﾄﾞﾗｲﾊﾞｲﾝｽﾄｰﾙ

ｽﾃｯﾌﾟ 1：付属の CD-ROM から PPCI7443 を選択して下さい。
ｽﾃｯﾌﾟ 2：画面の指示に従ってｲﾝｽﾄｰﾙして下さい。
ｽﾃｯﾌﾟ 3：ｲﾝｽﾄｰﾙ終了後、ﾊﾟｿｺﾝを再起動して下さい。

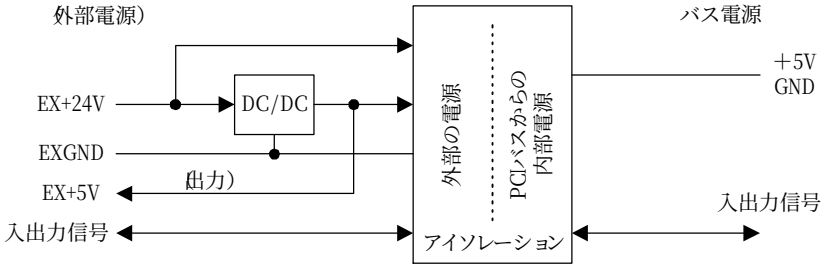
2.5 CN1ﾋﾟﾝ番号： 外部電源入力

CN1 Pin No.	名称	記述
1	EXGND	外部電源ＧＮＤ
2	EX+24V	外部電源 DC+24V ±5%

注意:

1. CN1 はねじのないﾌﾟﾗｸﾞｲﾝｺﾈｸﾀである。
2. 必ず外部電源を使用して下さい。DC+24V は外部の入力/出力信号に使用されます。電源回路は下記の通りに構成されます。
3. CN1 との接続のための推奨電線
   単線: φ0.32mm ～ φ0.65mm (AWG22 ～ AWG28)
   電線: 0.08mm2 ～ 0.32mm2 (AWG22 ～ AWG28)

下記の図は PPCI7443 の外部電源ｼｽﾃﾑを示します。外部の+24V 電源は必ず供給されなければなりません。基板上のﾚｷﾞｭﾚｰﾀ(DC/DC)が外部入出力用電源と内部用の+5V を作り出しています。



2.6　CN2ﾋﾟﾝ番号 :　ﾒｲﾝｺﾈｸﾀ

CN2 は制御入出力信号用のﾒｲﾝｺﾈｸﾀです。

No.	名称	I/O	機能 (軸番号)	No.	名称	I/O	機能 (軸番号)
1	VPP	O	+5V 電源出力	51	VPP	O	+5V 電源出力
2	GND		Ext. 電源 GND	52	GND		Ext. 電源 GND
3	OUT1+	O	ﾊﾟﾙｽ信号(+) ①	53	OUT3+	O	ﾊﾟﾙｽ信号(+) ③
4	OUT1	O	ﾊﾟﾙｽ信号(-) ①	54	OUT3	O	ﾊﾟﾙｽ信号(-) ③
5	DIR1+	O	方向信号(+) ①	55	DIR3+	O	方向信号(+) ③
6	DIR1	O	方向信号(-) ①	56	DIR3	O	方向信号(-) ③
7	SVON1	O	ｻｰﾎﾞ ON 信号 ①	57	SVON3	O	ｻｰﾎﾞ ON 信号 ③
8	ERC1	O	ｸﾘｱｰ信号 ①	58	ERC3	O	ｸﾘｱｰ信号 ③
9	ALM1	I	ｱﾗｰﾑ信号 ①	59	ALM3	I	ｱﾗｰﾑ信号 ③
10	INP1	I	ｲﾝﾎﾟｼﾞｼｮﾝ信号 ①	60	INP3	I	ｲﾝﾎﾟｼﾞｼｮﾝ信号 ③
11	RDY1	I	ﾚﾃﾞｨ信号 ①	61	RDY3	I	ﾚﾃﾞｨ信号 ③
12	GND		Ext. 電源 GND	62	EXGND		Ext. 電源 GND
13	EA1+	I	ｴﾝｺｰﾀﾞ A 相(+) ①	63	EA3+	I	ｴﾝｺｰﾀﾞ A 相(+) ③
14	EA1	I	ｴﾝｺｰﾀﾞ A 相(-) ①	64	EA3	I	ｴﾝｺｰﾀﾞ A 相(-) ③
15	EB1+	I	ｴﾝｺｰﾀﾞ B 相(+) ①	65	EB3+	I	ｴﾝｺｰﾀﾞ B 相(+) ③
16	EB1	I	ｴﾝｺｰﾀﾞ B 相(-) ①	66	EB3	I	ｴﾝｺｰﾀﾞ B 相(-) ③
17	EZ1+	I	ｴﾝｺｰﾀﾞ Z 相(+) ①	67	EZ3+	I	ｴﾝｺｰﾀﾞ Z 相(+) ③
18	EZ1	I	ｴﾝｺｰﾀﾞ Z 相(-) ①	68	EZ3	I	ｴﾝｺｰﾀﾞ Z 相(-) ③
19	VPP	O	+5V 電源出力	69	VPP	O	+5V 電源出力
20	GND		Ext. 電源 GND	70	GND		Ext. 電源 GND
21	OUT2+	O	ﾊﾟﾙｽ信号(+) ②	71	OUT4+	O	ﾊﾟﾙｽ信号(+) ④
22	OUT2	O	ﾊﾟﾙｽ信号(-) ②	72	OUT4	O	ﾊﾟﾙｽ信号(-) ④
23	DIR2+	O	方向信号(+) ②	73	DIR4+	O	方向信号(+) ④
24	DIR2	O	方向信号(-) ②	74	DIR4	O	方向信号(-) ④
25	SVON2	O	ｻｰﾎﾞ ON 信号 ②	75	SVON4	O	ｻｰﾎﾞ ON 信号 ④
26	ERC2	O	ｸﾘｱｰ信号 ②	76	ERC4	O	ｸﾘｱｰ信号 ④
27	ALM2	I	ｱﾗｰﾑ信号 ②	77	ALM4	I	ｱﾗｰﾑ信号 ④
28	INP2	I	ｲﾝﾎﾟｼﾞｼｮﾝ信号 ②	78	INP4	I	ｲﾝﾎﾟｼﾞｼｮﾝ信号 ④
29	RDY2	I	ﾚﾃﾞｨ信号 ②	79	RDY4	I	ﾚﾃﾞｨ信号 ④
30	GND		Ext. 電源 GND	80	GND		Ext. 電源 GND
31	EA2+	I	ｴﾝｺｰﾀﾞ A 相(+) ②	81	EA4+	I	ｴﾝｺｰﾀﾞ A 相(+) ④
32	EA2	I	ｴﾝｺｰﾀﾞ A 相(-) ②	82	EA4	I	ｴﾝｺｰﾀﾞ A 相(-) ④
33	EB2+	I	ｴﾝｺｰﾀﾞ B 相(+) ②	83	EB4+	I	ｴﾝｺｰﾀﾞ B 相(+) ④
34	EB2	I	ｴﾝｺｰﾀﾞ B 相(-) ②	84	EB4	I	ｴﾝｺｰﾀﾞ B 相(-) ④
35	EZ2+	I	ｴﾝｺｰﾀﾞ Z 相(+) ②	85	EZ4+	I	ｴﾝｺｰﾀﾞ Z 相(+) ④
36	EZ2	I	ｴﾝｺｰﾀﾞ Z 相(-) ②	86	EZ4	I	ｴﾝｺｰﾀﾞ Z 相(-) ④
37	+EL1	I	＋側ｴﾝﾄﾞﾘﾐｯﾄ信号 ①	87	+EL3	I	＋側ｴﾝﾄﾞﾘﾐｯﾄ信号 ③
38	-EL1	I	－側ｴﾝﾄﾞﾘﾐｯﾄ信号 ①	88	-EL3	I	－側ｴﾝﾄﾞﾘﾐｯﾄ信号 ③
39	CMP1	O	位置比較信号 ①	89	LTC3	I	位置ﾗｯﾁ信号 ③
40	SD/PCS1	I	ｽﾛｰﾀﾞｳﾝ/位置変更信号 ①	90	SD/PCS3	I	ｽﾛｰﾀﾞｳﾝ/位置変更信号 ③
41	ORG1	I	原点信号 ①	91	ORG3	I	原点信号 ③
42	GND		Ext. 電源 GND	92	GND		Ext. 電源 GND
43	+EL2	I	＋側ｴﾝﾄﾞﾘﾐｯﾄ信号 ②	93	+EL4	I	＋側ｴﾝﾄﾞﾘﾐｯﾄ信号 ④
44	-EL2	I	－側ｴﾝﾄﾞﾘﾐｯﾄ信号 ②	94	-EL4	I	－側ｴﾝﾄﾞﾘﾐｯﾄ信号 ④
45	CMP2	O	位置比較信号 ②	95	LTC4	I	位置ﾗｯﾁ信号 ④
46	SD/PCS2	I	ｽﾛｰﾀﾞｳﾝ/位置変更信号 ②	96	SD/PCS4	I	ｽﾛｰﾀﾞｳﾝ/位置変更信号 ④
47	ORG2	I	原点信号 ②	97	ORG4	I	原点信号 ④
48	GND		Ext. 電源 GND	98	GND		Ext. 電源 GND
49	GND		Ext. 電源 GND	99	E_24V	O	Ext. 電源、+24V
50	GND		Ext. 電源 GND	100	E_24V	O	Ext. 電源、+24V

2.7　CN3ﾋﾟﾝ番号：　ﾊﾟﾙｻｰ入力

CN3 は手動ﾊﾟﾙｻｰ入力用のｺﾈｸﾀです。

No.	名称	機能（軸番号）
1	GND	バス電源ＧＮＤ
2	PB4	パルサーＢ相信号入力　④
3	PA4	パルサーＡ相信号入力　④
4	PB3	パルサーＢ相信号入力　③
5	PA3	パルサーＡ相信号入力　③
6	+5V	バス電源＋５Ｖ
7	GND	バス電源ＧＮＤ
8	PB2	パルサーＢ相信号入力　②
9	PA2	パルサーＡ相信号入号　②
10	PB1	パルサーＢ相信号入力　①
11	PA1	パルサーＡ相信号入力　①
12	+5V	バス電源＋５Ｖ

注意:＋５ＶとＧＮＤピンはPCI バス電源によって直接与えられます。
したがって、これらの信号はｱｲｿﾚｰｼｮﾝされていません。

2.8　CN4ﾋﾟﾝ番号：　同時START/STOP

CN4 は複数枚の複数軸を同時に START/STOP するためのｺﾈｸﾀです。

No.	名称	機能（軸番号）
1	GND	バス電源ＧＮＤ
2	STP	同時の停止信号　入力/出力
3	STA	同時のスタート信号　入力/出力
4	STP	同時の停止信号　入力/出力
5	STA	同時のスタート信号　入力/出力
6	+5V	バス電源＋５Ｖ

注意:＋５ＶとＧＮＤピンはPCI バス電源によって直接与えられます。
したがって、これらの信号はｱｲｿﾚｰｼｮﾝされていません。

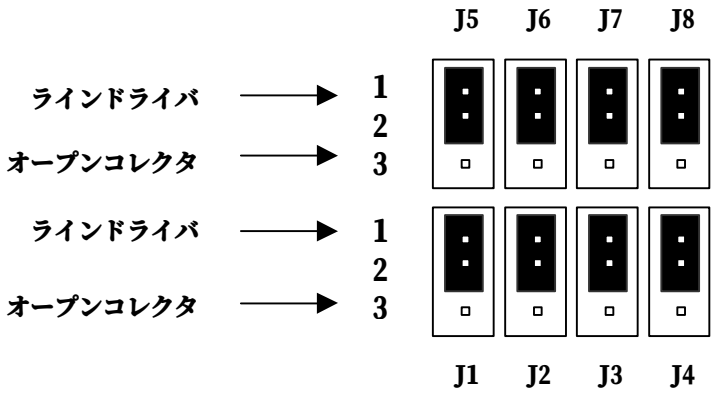
2.9　CN5ﾋﾟﾝ番号：　汎用TTL出力

CN5 は汎用出力(TTL 出力)用のｺﾈｸﾀです。

Pin No.	名前	機能
1	DGND	ﾃﾞｼﾞﾀﾙ GND
2	DGND	ﾃﾞｼﾞﾀﾙ GND
3	ED0	ﾃﾞｼﾞﾀﾙ出力 0
4	ED1	ﾃﾞｼﾞﾀﾙ出力 1
5	ED2	ﾃﾞｼﾞﾀﾙ出力 2
6	ED3	ﾃﾞｼﾞﾀﾙ出力 3
7	ED4	ﾃﾞｼﾞﾀﾙ出力 4
8	ED5	ﾃﾞｼﾞﾀﾙ出力 5
9	VCC	VCC +5V
10	N.C.	未使用

2.10　ｼﾞｬﾝﾊﾟｰの設定

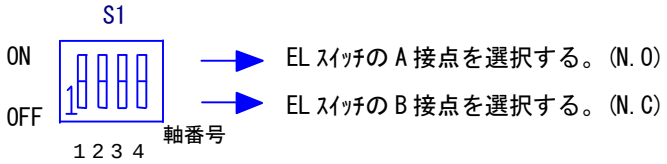
J1～J8 は、ﾊﾟﾙｽ出力信号のﾀｲﾌﾟを設定するのに使用されます(DIR ﾋﾟﾝと OUT ﾋﾟﾝ)。出力信号ﾀｲﾌﾟにはﾗｲﾝﾄﾞﾗｲﾊﾞ出力とｵｰﾌﾟﾝｺﾚｸﾀ出力があります。ｼﾞｬﾝﾊﾟｰ設定の詳細についてはｾｸｼｮﾝ 3.1 をご参照下さい。初期設定はﾗｲﾝﾄﾞﾗｲﾊﾞ出力ﾀｲﾌﾟになっています。



2.11　ﾃﾞｨｯﾌﾟｽｲｯﾁの設定

ｽｲｯﾁＳ１は、EL のﾀｲﾌﾟを設定するのに使用されます。EL のﾀｲﾌﾟの初期設定はﾉｰﾏﾙｵｰﾌﾟﾝﾀｲﾌﾟです（Ａ接点）。ｽｲｯﾁを OFF にすると、ﾉｰﾏﾙｸﾛｰｽﾞﾀｲﾌﾟになります（Ｂ接点）。初期設定はﾉｰﾏﾙｵｰﾌﾟﾝﾀｲﾌﾟに設定されています。

ﾎﾞｰﾄﾞ上の S1 ｽｲｯﾁの意味



3

信号接続

すべての入出力信号の信号接続は本章で説明されます。PPCI7443 とﾓｰﾀﾄﾞﾗｲﾊﾞを配線する前に、本章の内容をよく理解して下さい。

本章は下記のセクションから構成されます。

セクション 3.1	ﾊﾟﾙｽ出力信号	OUT, DIR
セクション 3.2	ｴﾝｺｰﾀﾞﾌｨｰﾄﾞﾊﾞｯｸ信号	EA, EB, EZ
セクション 3.3	原点信号	ORG
セクション 3.4	ｴﾝﾄﾞﾘﾐｯﾄ信号	+EL, -EL
セクション 3.5	ｽﾛｰﾀﾞｳﾝ信号と PCS 信号	SD/PCS
セクション 3.6	ｲﾝﾎﾟｼﾞｼｮﾝ信号	INP
セクション 3.7	ｱﾗｰﾑ信号	ALM
セクション 3.8	偏差ｶｳﾝﾀｸﾘｱ信号	ERC
セクション 3.9	ｻｰﾎﾞ ON 信号(汎用出力信号)	SVON
セクション 3.10	ﾚﾃﾞｨ信号(汎用入力信号)	RDY
セクション 3.11	位置比較出力信号	CMP
セクション 3.12	位置ﾗｯﾁ入力信号	LTC
セクション 3.13	ﾊﾟﾙｻｰ入力信号	PA, PB
セクション 3.14	同時 START/STOP 信号	STA, STP
セクション 3.15	汎用の TTL 出力信号	ED

3.1　ﾊﾟﾙｽ出力信号　OUT, DIR

４軸のパルス出力が PPCI7443 にあります。各軸は、OUT 信号と DIR 信号が対であり、方向信号と連続したﾊﾟﾙｽを出力します。OUT 信号と DIR 信号はﾌﾟﾛｸﾞﾗﾑにより CW 信号と CCW 信号のペアに設定することができます。OUT 信号、 DIR 信号 の論理的な特徴の詳細はｾｸｼｮﾝ 4.1.1 をご参照下さい。このｾｸｼｮﾝでは、OUT 信号と DIR 信号の電子特性を示します。各信号は１組の差動信号から成ります。例えば、OUT2 では OUT2+と OUT2-からなっているということです。下記の表は CN2 のすべてのﾊﾟﾙｽ出力信号を表示します。

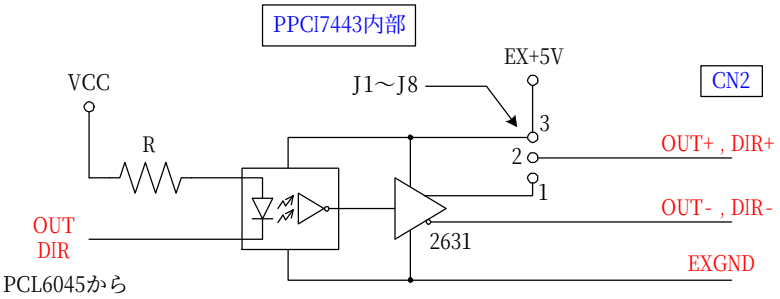
CN2 Pin No.	信号名	機能	軸番号
3	**OUT1+**	パルス信号(+)	①
4	**OUT1-**	パルス信号(-)	①
5	**DIR1+**	方向信号(+)	①
6	**DIR1-**	方向信号(-)	①
21	**OUT2+**	パルス信号(+)	②
22	**OUT2-**	パルス信号(-)	②
23	**DIR2+**	方向信号(+)	②
24	**DIR2-**	方向信号(-)	②
53	**OUT3+**	パルス信号(+)	③
54	**OUT3-**	パルス信号(-)	③
55	**DIR3+**	方向信号(+)	③
56	**DIR3-**	方向信号(-)	③
71	**OUT4+**	パルス信号(+)	④
72	**OUT4-**	パルス信号(-)	④
73	**DIR4+**	方向信号(+)	④
74	**DIR4-**	方向信号(-)	④

OUT 信号、DIR 信号は、ｼﾞｬﾝﾊﾟｰによりﾗｲﾝﾄﾞﾗｲﾊﾞ出力かｵｰﾌﾟﾝｺﾚｸﾀ出力のどちらかを選択することができます。
下記のｼﾞｬﾝﾊﾟｰ J1～J8 の設定を１と２を短絡するか２と３を短絡するかによって、出力ﾓｰﾄﾞを選択することができます。

出力信号	ﾗｲﾝﾄﾞﾗｲﾊﾞ出力 １－２短絡	ｵｰﾌﾟﾝｺﾚｸﾀ出力 ２－３短絡
OUT1	J1	J1
DIR1	J2	J2
OUT2	J3	J3
DIR2	J4	J4
OUT3	J5	J5
DIR3	J6	J6
OUT4	J7	J7
DIR4	J8	J8

初期設定は OUT 信号、DIR 信号共にﾗｲﾝﾄﾞﾗｲﾊﾞ出力です。

下記の図は４軸に関する OUT 信号と DIR 信号の内部回路です。



注意：ﾊﾟﾙｽ出力がｵｰﾌﾟﾝｺﾚｸﾀ出力ﾓｰﾄﾞに設定されている場合は、OUT-信号と DIR-信号を出力するのに使用します。また、OUT-と DIR-ﾋﾟﾝへのｼﾝｸ電流が 20mA を超えないように注意して下さい。電流は EX+5V 電源で供給されます。但し、EX+5V 電源の最大容量が 500mA であるのに注意して下さい。

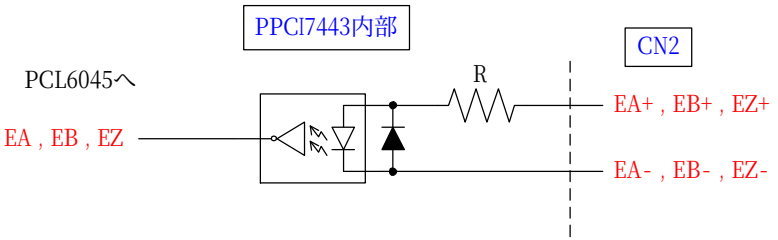
3.2 エンコーダフィードバック信号 EA, EB, EZ

エンコーダフィードバック信号には EA、EB、および EZ 信号があります。各軸には、それぞれ３組のＡ相(EA)、Ｂ相(EB)、およびＺ相(EZ)入力のための 6 本のピンがあります。EA と EB は位置のカウントに使用されて、EZ はゼロ位置に使用されます。相対的な信号名、ピン番号、および軸の番号は下記の表に示されます。

CN2 Pin No.	信号名	軸番号	CN2 Pin No.	信号名	軸番号
13	EA1+	①	63	EA3+	③
14	EA1−	①	64	EA3−	③
15	EB1+	①	65	EB3+	③
16	EB1−	①	66	EB3−	③
31	EA2+	②	81	EA4+	④
32	EA2−	②	82	EA4−	④
33	EB2+	②	83	EB4+	④
34	EB2−	②	84	EB4−	④

CN2Pin No.	信号名	軸番号	CN2Pin No.	信号名	軸番号
17	EZ1+	①	67	EZ3+	③
18	EZ1−	①	68	EZ3−	③
35	EZ2+	②	85	EZ4+	④
36	EZ2−	②	86	EZ4−	④

EA、EB、EZ 信号の内部回路は下記の通りです。

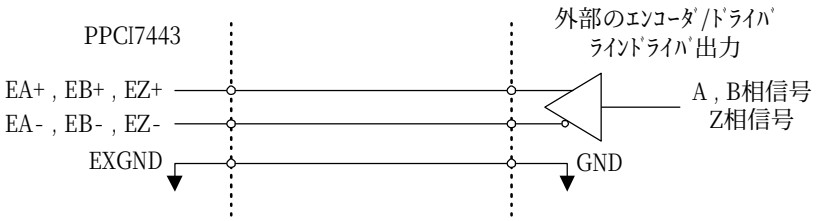


すべてのエンコーダ入力信号(EA+, EA−, EB+, EB−, EZ+, EZ−)は少なくとも 3.5V 以上必要であるのに注意して下さい。そのため、エンコーダフィードバックもしくはモータドライバフィードバックを接続するときには、動作能力を確かめる必要があります。異なる符号の信号の対は、PCL6045 に接続するためにデジタル信号 EA、EB、および EZ に変換されます。

ここに、ｴﾝｺｰﾀﾞ入力信号の接続例を２例説明します。入力回路はｴﾝｺｰﾀﾞかﾓｰﾀﾄﾞﾗｲﾊﾞに接続することができ、一つがﾗｲﾝﾄﾞﾗｲﾊﾞ出力の時でもう一つがｵｰﾌﾟﾝｺﾚｸﾀ出力の時の例です。

◆ ﾗｲﾝﾄﾞﾗｲﾊﾞ出力との接続

PPCI7443 ｴﾝｺｰﾀﾞ入力をﾄﾞﾗｲﾌﾞするために、ﾄﾞﾗｲﾊﾞ出力は少なくとも 6mA ﾄﾞﾗｲﾌﾞできる能力の差動ペア線に少なくとも 3.5V 供給する必要があります。また、両サイドの GND は共通にして下さい。

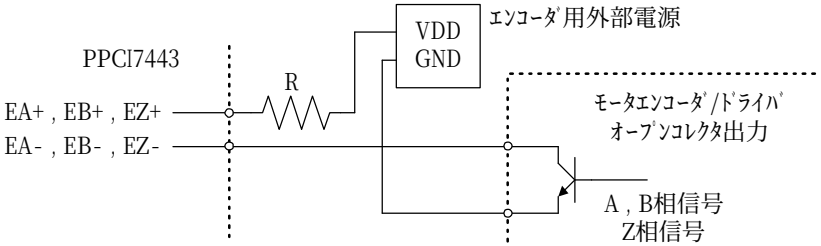


◆ ｵｰﾌﾟﾝｺﾚｸﾀ出力との接続

ｵｰﾌﾟﾝｺﾚｸﾀ出力との接続には外部の電源が必要です。PPCI7443 と、ｴﾝｺｰﾀﾞと、電源との接続は下記の図で示されます。PPCI7443 の入力回路の保護用に、外部に制限低抗Ｒを入れるのに注意して下さい。下記の表は、ｴﾝｺｰﾀﾞ電源によって入れる制限抵抗の値を示します。

ｴﾝｺｰﾀﾞ電源(VDD)	制限低抗　R
+5V	0 (なし)
+12V	1.8kΩ
+24V	4.3kΩ

If=6mA Max



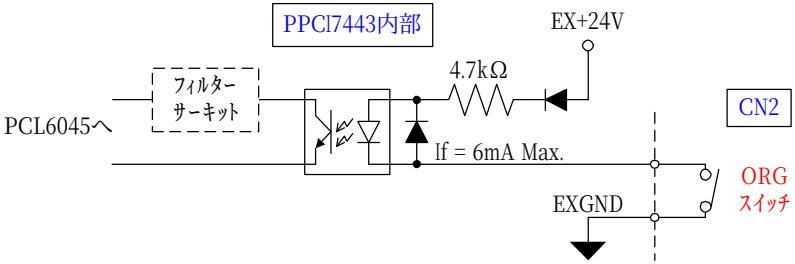
ｴﾝｺｰﾀﾞﾌｨｰﾄﾞﾊﾞｯｸ信号のその他の詳細操作については、ｾｸｼｮﾝ 4.4 をご参照下さい。

3.3 原点信号 ORG

原点信号(ORG1～ORG4)は機械的なｽｲｯﾁの入力信号として使用されます。相対的な信号名、ﾋﾟﾝ番号、および軸の番号は下記の表に示されます。

CN2 Pin No.	信号名	軸番号
41	ORG1	①
47	ORG2	②
91	ORG3	③
97	ORG4	④

原点信号の内部回路は下記のように示されます。通常、ORG ｽｲｯﾁは、軸の原点を決めるのに使用されます。ORG ｽｲｯﾁの仕様は DC+24V の最小接触容量 6mA のものを使用して下さい。内部のﾌｨﾙﾀ回路は、高周波ｽﾊﾟｲｸを除去するのに使用されます。(ｽﾊﾟｲｸは誤動作を引き起こすかもしれない)。



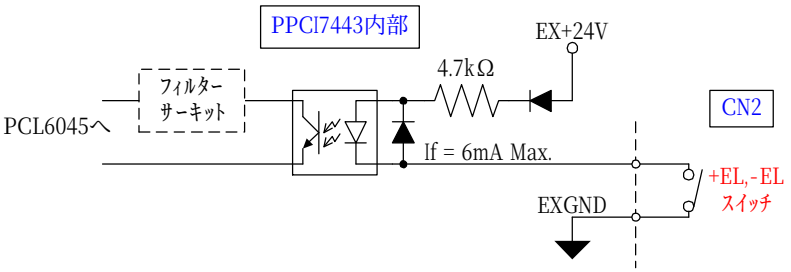
軸の制御が原点復帰ﾓｰﾄﾞで操作されるとき、ORG 信号は、ﾊﾟﾙｽ出力信号(OUT と DIR)を止めるのに使用されます。ORG の詳細操作についてはｾｸｼｮﾝ4.3.3 をご参照下さい。

3.4　エンドリミット信号　+EL, -EL

各軸に＋EL と－EL の２つエンドリミット信号があります。＋EL は＋側エンドリミットを表し－EL は－側のエンドリミット信号を表します。相対的な信号名、ピン番号、および軸の番号は下記の表に示されます。

CN2Pin No.	信号名	軸番号	CN2Pin No.	信号名	軸番号
37	+EL1	①	87	+EL3	③
38	-EL1	①	88	-EL3	③
43	+EL2	②	93	+EL4	④
44	-EL2	②	94	-EL4	④

エンドリミットとの接続する回路図は下記のような図で示されます。外部のリミットスイッチの仕様は DC+24V の最小接触容量 6mA のものを使用して下さい。ディップスイッチ(S1)を設定することによってＡ接点(ノーマルオープン)かＢ接点(ノーマルクローズ)かを設定することができます(セクション 2.9 参照)。出荷時の設定はＡ接点に設定してあります。EL 操作の詳細についてはセクション 4.3.2 をご参照下さい。

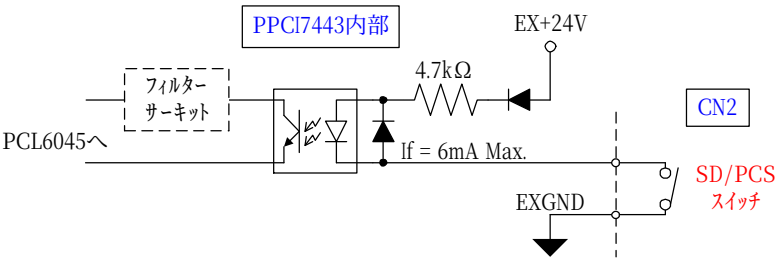


3.5　ｽﾛｰﾀﾞｳﾝ信号とPCS信号　SD/PCS

各軸すべてに SD/PCS 信号がそれぞれあります。相対的な信号名、ﾋﾟﾝ番号、および軸の番号は下記の表に示されます。

CN2 Pin No.	信号名	軸番号
40	SD1/PCS1	①
46	SD2/PCS2	②
90	SD3/PCS3	③
96	SD4/PCS4	④

この信号の内部回路が下記の図で示されます。通常、ｽﾛｰﾀﾞｳﾝｽｲｯﾁは、ﾓｰﾀの速度における操作で高速から低速にするための信号を発生させるのに使用されます。SD/PCS 操作に関するその他の詳細については、ｾｸｼｮﾝ 4.3.1 をご参照下さい。

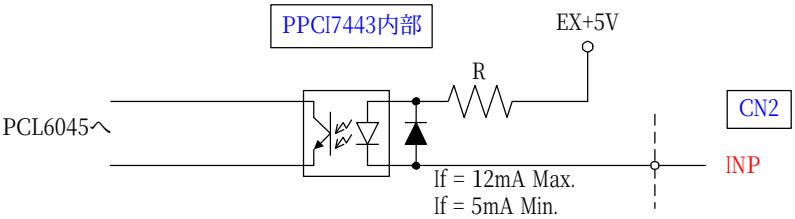


3.6　インポジション信号　INP

サーボモータドライバからのインポジション信号(INP)は、偏差カウンタが「0」であることを示します。すなわちサーボモータの動作完了を意味します。相対的な信号名、ピン番号、および軸の番号は下記の表に示されます。

CN2Pin No.	信号名	軸番号
10	INP1	①
28	INP2	②
60	INP3	③
78	INP4	④

INP 信号の入力回路は下記の図で見せられます。



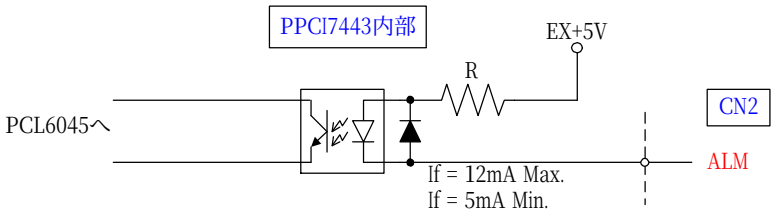
インポジション信号は、通常サーボドライバから入力されます(通常、サーボドライバはオープンコレクタ出力信号)。外部回路は、INP 信号を動作させるために少なくとも 5mA 以上のシンク電流を供給する必要があります。INP 信号動作に関するその他の詳細についてはセクション 4.2.1 をご参照下さい。

3.7 ｱﾗｰﾑ信号 ALM

ｱﾗｰﾑ信号 ALM は、ｻｰﾎﾞﾄﾞﾗｲﾊﾞからｱﾗｰﾑ状態を示すのに使用されます。相対的な信号名、ﾋﾟﾝ番号、および軸の番号は下記の表に示されます。

CN2 Pin No.	信号名	軸番号
9	ALM1	①
27	ALM2	②
59	ALM3	③
77	ALM4	④

ｱﾗｰﾑ信号の入力回路は下記の図で示されます。ALM 信号は通常ｻｰﾎﾞﾄﾞﾗｲﾊﾞより入力されます(通常、ｻｰﾎﾞﾄﾞﾗｲﾊﾞはｵｰﾌﾟﾝｺﾚｸﾀ出力)。外部回路は、ALM 信号を動作させるために少なくとも 5mA 以上のｼﾝｸ電流を供給する必要があります。ALM 信号の操作に関するその他の詳細についてはｾｸｼｮﾝ 4.2.2 をご参照下さい。



3.8　偏差ｶｳﾝﾀｸﾘｱ信号 ERC

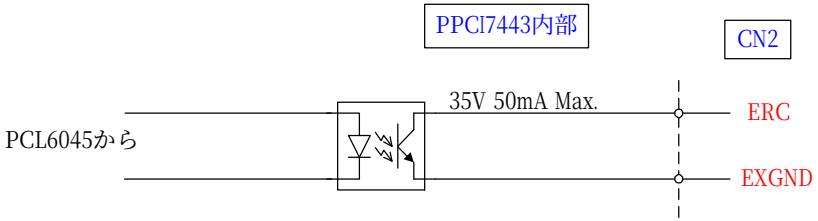
偏差ｶｳﾝﾀｸﾘｱ信号(ERC)は下記の４つの状況で出力されます。（ERC 信号が有効に設定されている場合）

(1) 原点復帰完了
(2) ｴﾝﾄﾞﾘﾐｯﾄｽｲｯﾁが ON した
(3) ｱﾗｰﾑ信号による OUT と DIR 信号の停止
(4) 非常停止ｺﾏﾝﾄﾞがｿﾌﾄｳｪｱ(ｵﾍﾟﾚｰﾀ)によって発行された

相対的な信号名、ﾋﾟﾝ番号、および軸の番号は下記の表に示されます。

CN2 Pin No.	信号名	軸番号
8	ERC1	①
26	ERC2	②
58	ERC3	③
76	ERC4	④

ERC 信号は、ｻｰﾎﾞﾄﾞﾗｲﾊﾞの偏差ｶｳﾝﾀを「0」にするのに使用されます。ERC 出力回路は外部電源、最高 35V、50mA を駆動する能力のｵｰﾌﾟﾝｺﾚｸﾀ出力回路です。ERC 操作に関するその他の詳細についてはｾｸｼｮﾝ 4.2.3 をご参照下さい。

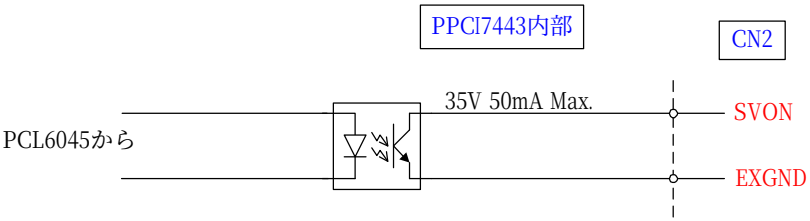


3.9　サーボオン信号　　SVON

SVON 信号はｻｰﾎﾞﾓｰﾀｵﾝ信号か汎用出力信号として使用できます。相対的な信号名、ﾋﾟﾝ番号、および軸の番号は下記の表に示されます。

CN2 Pin No.	信号名	軸番号
7	SVON1	①
25	SVON2	②
57	SVON3	③
75	SVON4	④

SVON 信号の内部回路は下記の図で見せられます。

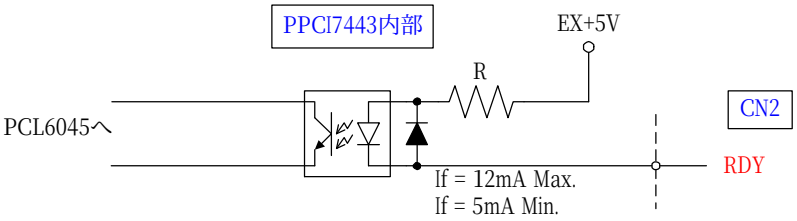


3.10　ﾚﾃﾞｨ信号　　RDY

RDY 信号はﾓｰﾀﾄﾞﾗｲﾊﾞのﾚﾃﾞｨ信号か汎用入力信号として使用できます。
汎用入力としても使用できるのでｻｰﾎﾞﾄﾞﾗｲﾊﾞｰからのﾚﾃﾞｨ信号が OFF でもﾊﾟﾙｽ出力を自動的に停止させることはありません。相対的な信号名、ﾋﾟﾝ番号、および軸の番号は下記の表に示されます。

CN2 Pin No.	信号名	軸番号
11	RDY1	①
29	RDY2	②
61	RDY3	③
79	RDY4	④

RDY 信号の内部回路は下記の図で見せられます。

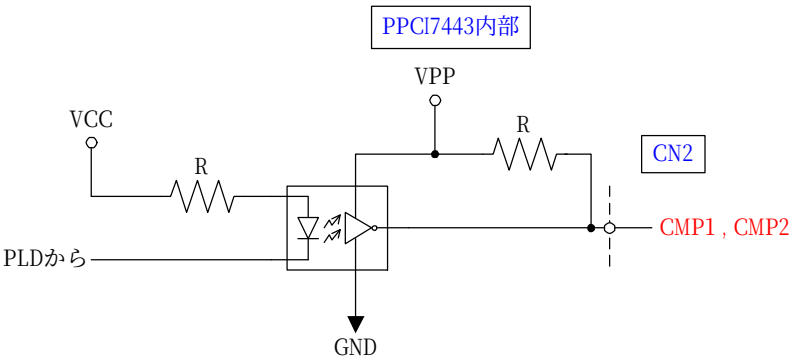


3.11　比較出力信号　　CMP

PPCI7443 は位置比較ﾄﾘｶﾞｰ出力として２個の CMP 信号があります。それらは最初の２軸（①と②）のみ有効でです。ｴﾝｺｰﾀﾞｶｳﾝﾀがﾕｰｻﾞｰによって事前に設定された値に達したとき、比較出力はﾊﾟﾙｽ信号を発生させます。CMP 信号は CN2 にあります。相対的な信号名、ﾋﾟﾝ番号、および軸の番号は下記の表に示されます。

CN2Pin No.	信号名	軸番号
39	CMP1	①
45	CMP2	②

下記の図は最初の２軸の CMP 信号の内部回路です。



注意

ＣＭＰﾄﾘｶﾞｰ出力は立ち上がりエッジか立ち下がりエッジの設定ができます。初期値は立ち下がりエッジになっています。詳しくはセクション 6.16 の _7443_set_trigger_type()を参照して下さい。
このＣＭＰピン出力はＴＴＬ出力と見なします。

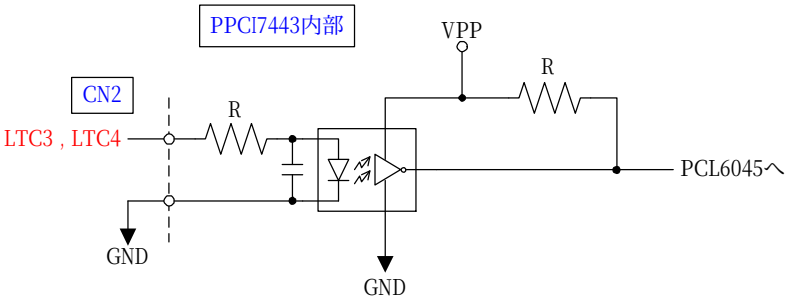
上記の図において
ＶＰＰ：アイソレーション５Ｖ
ＶＣＣ：コンピュータ５Ｖ
Ｒ１：４７０Ω
Ｒ２：１ｋΩ

3.12　位置ﾗｯﾁ入力信号　　LTC

PPCI7443 は位置ﾗｯﾁ入力として２個の LTC 信号を用意してあります。それらは後ろの２軸（③と④）のみ有効です。LTC 信号はｶｳﾝﾀ値をﾗｯﾁするﾄﾘｶﾞｰであり、そしてそれは、正確に位置を確定します。
LTC 信号は CN2 にあります。相対的な信号名、ﾋﾟﾝ番号、および軸の番号は下記の表に示されます。

CN2 Pin No.	信号名	軸番号
89	LTC3	③
95	LTC4	④

下記の図は最後の２軸の LTC 信号の内部回路です。

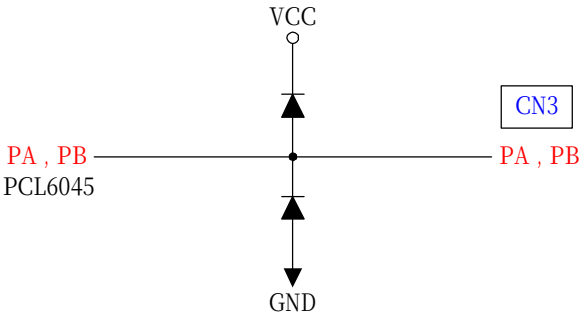


3.13　ﾊﾟﾙｻｰ入力信号　PA, PB

PPCI7443 はｺﾈｸﾀ(CN3)よりﾊﾟﾙｻｰ信号を入力することができます。ﾊﾟﾙｻｰの波形はｴﾝｺｰﾀﾞと同様の波形です。この信号は通常、位置情報(ﾊﾟﾙｽ)を発生させるのに使用されます。それにﾓｰﾀが追従して動作します。

CN3 Pin No.	信号名	軸番号	CN3 Pin No.	信号名	軸番号
2	PA1	①	8	PA3	③
3	PB1	①	9	PB3	③
4	PA2	②	10	PA4	④
5	PB2	②	11	PB4	④

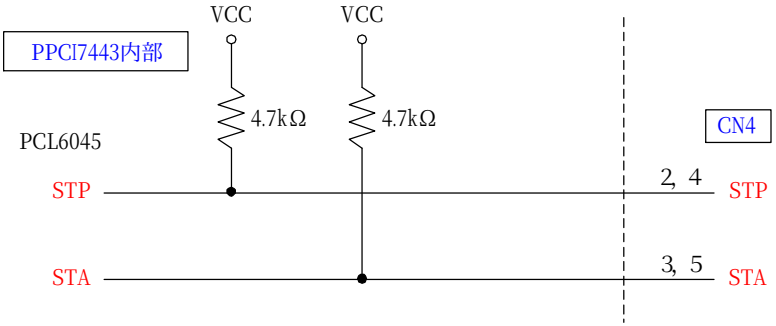
ｺﾈｸﾀ CN3 の PA と PB ﾋﾟﾝは直接 PCL6045 の PA と PB ピンに接続されています。内部回路は下記の通りに示されます。



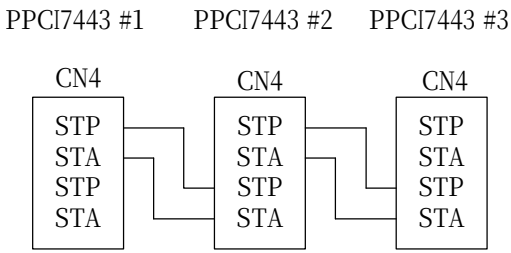
ﾊﾟﾙｻｰの信号電圧が+5V でないか、ﾊﾟﾙｻｰまでの距離が長い場合は、途中にﾌｫﾄｶﾌﾟﾗかﾗｲﾝﾄﾞﾗｲﾊﾞを入れることを勧めます。また、CN3 の+5V と GND は PCI バスから直接供給されますのでｱｲｿﾚｰｼｮﾝされていません。十分注意して使用して下さい。

3.14　同時START/STOP信号　　STA, STP

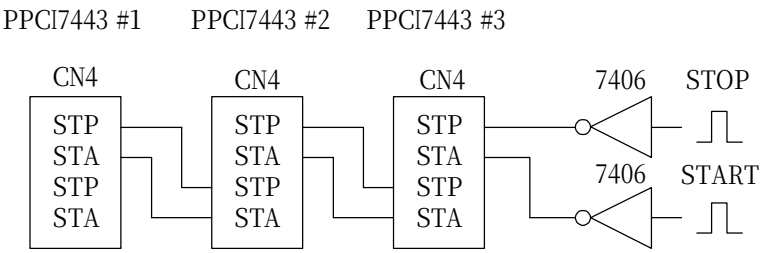
PPCI7443 は複数の基板の複数軸を同時にｽﾀｰﾄもしくはｽﾄｯﾌﾟするときに STA 信号と STP 信号によって行えます。(信号は複数軸における動作の同時の開始/停止を可能にする)。STA 信号と STP 信号は CN4 にあります。
下記の図は内部回路を示します。４軸の STA と STP 信号はそれぞれ接続されています。



STP 信号と STA 信号は入力と出力両方の信号です。同時 START/STOP 動作を操作するのに、ｿﾌﾄｳｪｱ制御と外部制御の両方が可能です。ｿﾌﾄｳｪｱ制御により、信号はどの PCL6045 からでも制御することができ、ｱﾌﾟﾘｹｰｼｮﾝﾌﾟﾛｸﾞﾗﾑが正しければ、他のﾁｯﾌﾟも同時に START/STOP することができます。
また、同時 START/STOP に関する STA/STP 信号を制御するのに外部のｵｰﾌﾟﾝｺﾚｸﾀかｽｲｯﾁを使用することでも操作可能です。
２枚以上の PPCI7443 で、同時の START/STOP を制御するすべての基板の CN4 ｺﾈｸﾀを接続したならば、関係する全ての軸が動作可能です。
この場合、下記のように CN4 を接続して下さい。



外部信号による同時 START/STOP を開始するには、7406(ｵｰﾌﾟﾝｺﾚｸﾀ)を使用した接続でも操作可能です。

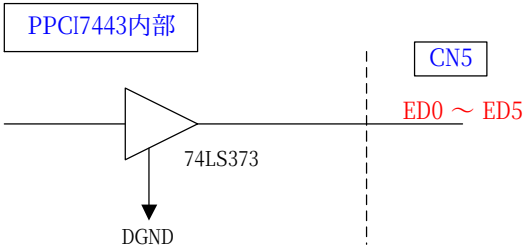


3.15 汎用のTTL出力信号 ED

PPCI7443 は６個の汎用 TTL ﾃﾞｼﾞﾀﾙ出力があります。TTL 出力は CN5 にあります。相対的な信号名、ﾋﾟﾝ番号、および軸の番号は下記の表に示されます。

Pin No.	名称	機能
1	DGND	ﾃﾞｼﾞﾀﾙ GND
2	DGND	ﾃﾞｼﾞﾀﾙ GND
3	ED0	ﾃﾞｼﾞﾀﾙ出力 0
4	ED1	ﾃﾞｼﾞﾀﾙ出力 1
5	ED2	ﾃﾞｼﾞﾀﾙ出力 2
6	ED3	ﾃﾞｼﾞﾀﾙ出力 3
7	ED4	ﾃﾞｼﾞﾀﾙ出力 4
8	ED5	ﾃﾞｼﾞﾀﾙ出力 5
9	VCC	VCC +5V

下記の図は TTL ﾃﾞｼﾞﾀﾙ出力の内部回路です。



4

操作原理

本章は PPCI7443 の詳細操作について説明します。下記にセクションの内容を記述します。

セクション 4.1: 動作制御モード
セクション 4.2: ﾓｰﾀﾄﾞﾗｲﾊﾞｲﾝﾀｰﾌｪｰｽ(INP, ERC, ALM, SVON, RDY)
セクション 4.3: ﾘﾐｯﾄｽｲｯﾁｲﾝﾀｰﾌｪｰｽと入出力状態(SD/PCS, EL, ORG)
セクション 4.4: ｴﾝｺｰﾀﾞﾌｨｰﾄﾞﾊﾞｯｸ信号(EA, EB, EZ)
セクション 4.5: 複数の PPCI7443 操作
セクション 4.6: ｵｰﾊﾞｰﾗｲﾄﾞ機能(速度、位置)
セクション 4.7: 位置の比較とラッチ機能
セクション 4.8: ﾊｰﾄﾞｳｪｱﾊﾞｯｸﾗｯｼｭ補正
セクション 4.9: ｿﾌﾄｳｪｱﾘﾐｯﾄ機能
セクション 4.10: 割り込み信号制御
セクション 4.11: ｱｲﾄﾞﾘﾝｸﾞ制御

4.1 動作制御モード

このセクションでは、ﾊﾟﾙｽ出力信号の構成と下記の動作制御ﾓｰﾄﾞについて説明します。

- 4.1.1 ﾊﾟﾙｽ出力ﾓｰﾄﾞ
- 4.1.2 １軸の一定速度動作
- 4.1.3 １軸の台形動作
- 4.1.4 １軸のＳ字曲線動作
- 4.1.5 ２～４軸の直線補間
- 4.1.6 ２軸の円弧補間

- 4.1.7　加減速付きの円弧補間
- 4.1.8　速度と加速時間の関係
- 4.1.9　連続動作
- 4.1.10　原点復帰動作
- 4.1.11　ﾊﾟﾙｻｰ動作
- 4.1.12　ﾀｲﾏｰﾓｰﾄﾞ
- 4.1.13　ﾊﾟﾙｻｰによる補間

4.1.1　パルス出力モード

PPCI7443 は、ﾄﾞﾗｲﾊﾞによりｻｰﾎﾞﾓｰﾀ/ｽﾃﾝﾋﾟﾝｸﾞﾓｰﾀを制御するのにﾊﾟﾙｽ指令を使用します。ﾊﾟﾙｽ指令は２つの信号から成ります。一つが OUT 信号でもう一つが DIR 信号です。この信号には２つのﾓｰﾄﾞがあります。一つがﾊﾟﾙｽ信号と方向信号の１ﾊﾟﾙｽ出力ﾓｰﾄﾞ(OUT/DIR)で、もう一つが＋ﾊﾟﾙｽ信号とｰﾊﾟﾙｽ信号の２ﾊﾟﾙｽ出力ﾓｰﾄﾞ(CW/CCW)の２種類です。
_7443_set_pls_outmode()ｺﾏﾝﾄﾞは、このﾊﾟﾙｽ出力ﾓｰﾄﾞを設定するのに使用されます。ﾓｰﾄﾞと OUT 信号と DIR 信号の対応表を下記に示します。

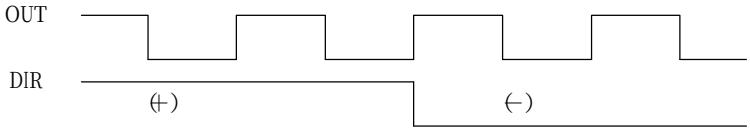
モード	OUT ピン	DIR ピン
２パルスモード	ＣＷ信号	ＣＣＷ信号
１パルスモード	パルス信号	方向信号（レベル）

これらの信号のｲﾝﾀｰﾌｪｰｽ回路はﾗｲﾝﾄﾞﾗｲﾊﾞかｵｰﾌﾟﾝｺﾚｸﾀ出力の選択が可能です。信号ﾀｲﾌﾟの選択はｼﾞｬﾝﾊﾟｰ設定により行います。設定方法はｾｸｼｮﾝ 3.1 をご参照下さい。

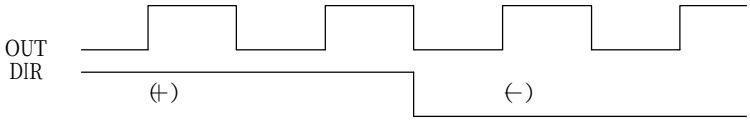
◆ １パルス出力モード(OUT/DIR)

このﾓｰﾄﾞでは、OUT 信号がﾊﾟﾙｽ指令を意味します。OUT 信号のﾊﾟﾙｽの数が相対的な「距離」または「位置」のための動作ｺﾏﾝﾄﾞを表し、OUT 信号のﾊﾟﾙｽ周波数が「ｽﾋﾟｰﾄﾞ」または「速度」のための動作指令を表します。 DIR 信号は正(＋)か負(－)の方向指令を表します。このﾓｰﾄﾞは最も一般的に使用されるﾓｰﾄﾞです。下記の図は出力波形を示します。

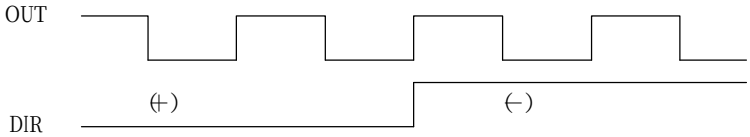
pls_outmode = 0 :



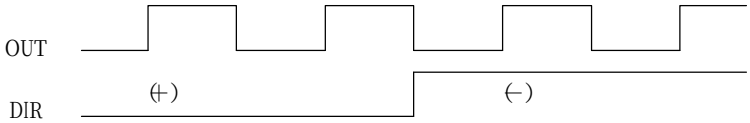
pls_outmode = 1 :



pls_outmode = 2 :



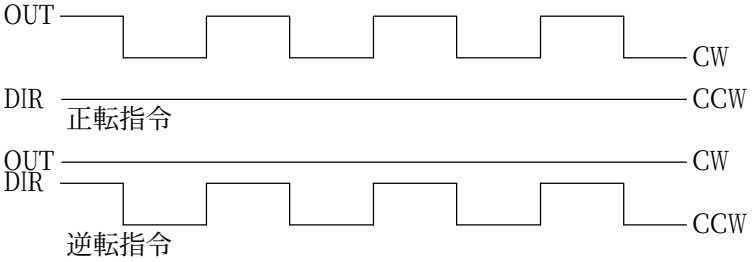
pls_outmode = 3 :



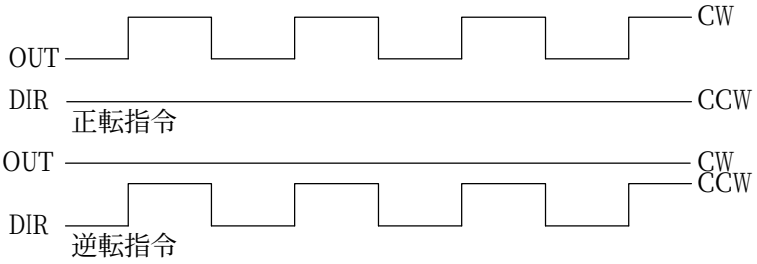
◆ ２パルス出力モード(CW/CCW)

このﾓｰﾄﾞでは、OUT と DIRﾋﾟﾝから出力される波形はそれぞれ CW(時計回り)と CCW(反時計回り)のﾊﾟﾙｽ出力を表します。OUTﾋﾟﾝ(CW)から出力されるﾊﾟﾙｽはﾓｰﾀを正転指令で動作させ、DIRﾋﾟﾝ(CCW)からの ﾊﾟﾙｽ出力はﾓｰﾀを逆転指令で動作させます。下記の図が正転指令(＋)と逆転指令(－)の出力波形を示します。

pls_outmode = 4 :



pls_outmode = 5 :



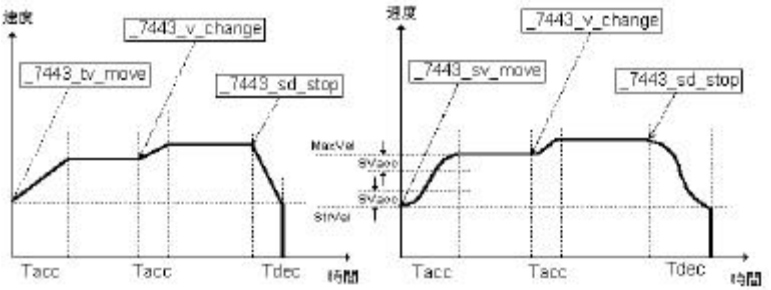
◆ 関連ﾌｧﾝｸｼｮﾝ：

_7443_set_pls_optmode()：セクション6.4参照

4.1.2　一定速度の動作

このﾓｰﾄﾞは、一定の速度で１軸のﾓｰﾀを動作させるのに使用されます。出力ﾊﾟﾙｽはｽﾀｰﾄ速度(***StrVel***)から指定された一定速度(***MaxVel***)まで加速します。この時の移動方向は MaxVel に与えられた値の符号によります。***_7443_tv_move()***ｺﾏﾝﾄﾞは台形による加速を行う時に使用し、***_7443_sv_move()***ｺﾏﾝﾄﾞはＳ字曲線にる加速を行う時に使用します。ﾊﾟﾙｽ出力速度は、別の速度ｺﾏﾝﾄﾞが実行されるか、停止ｺﾏﾝﾄﾞが実行されるまで最高速度(***MaxVel***)で保たれます。***_7443_v_change()***ｺﾏﾝﾄﾞは、ﾓｰﾀ動作中に速度を変えるのに使用されます。このﾌｧﾝｸｼｮﾝにより速度をより速くする場合には、実行する前に必ず***_7443_fix_speed_range()***ｺﾏﾝﾄﾞを呼び出して下さい。詳細説明についてｾｸｼｮﾝ 4.6 をご参照下さい。***_7443_sd_stop()***ｺﾏﾝﾄﾞは、ﾓｰﾀの速度をゼロに減速(停止)するのに使用されます。***_7443_emg_stop()***ｺﾏﾝﾄﾞは、ﾓｰﾀを急停止させるのに使用されます。それらの速度変化機能か停止機能は移動機能、***tv_move***または***sv_move***などの速度動作の後に続きます。この速度動作は下記の図のように示されます。

注意:***v_change***機能 と ***sd_stop***機能そして ***emg_stop***機能はﾌﾟﾘｾｯﾄﾓｰﾄﾞ（台形動作 4.1.3 参照、Ｓ字曲線動作 4.1.4 参照)か原点復帰ﾓｰﾄﾞ（4.1.10 参照）にも適用することができます。

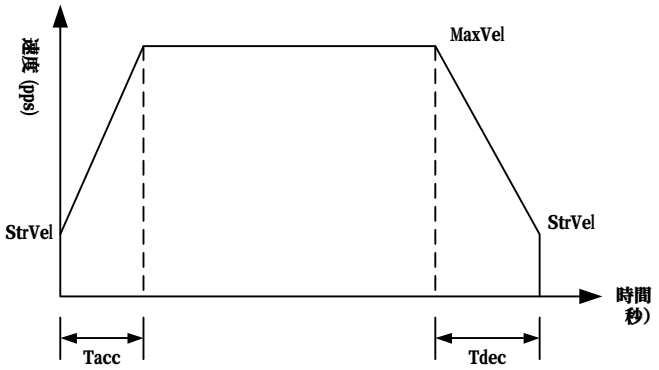


◆ 関連ﾌｧﾝｸｼｮﾝ:

_7443_tv_move(),_7443_sv_move(),_7443_v_change(),
_7443_sd_stop(),_7443_emg_stop(),_7443_fix_speed_range(),
_7443_unfix_speed_range() : セクション6.5参照

4.1.3　台形動作

このﾓｰﾄﾞは、台形の速度動作で指定された所定の位置(または距離)にﾓｰﾀを動作させるのに使用されます。ﾓｰﾀはある位置からある位置まで制御されます。ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞまたはｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでの動作ができます。ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでは目標位置が座標で指定され、ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでは目標位置までの距離(ﾊﾟﾙｽ数)が指定されます。ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞとｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞ両方共に、加速と減速が異なって設定できます。***_7443_motion_done()***ｺﾏﾝﾄﾞは、軸の動作状態をﾁｪｯｸするのに使用されます。
下記の図は台形動作を示します。



PPCI7443 にはある位置からある位置への台形動作に２つあります。
_7443_start_ta_move()ｺﾏﾝﾄﾞでは、ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでの目標位置がﾊﾟﾙｽの単位で与えられなければなりません。１回の動作の物理的な長さや角度はﾓｰﾀﾄﾞﾗｲﾊﾞと機械の構造(ﾓｰﾀを含んでいる)に依存しています。ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでは実際の現在位置が必要ですので、***_7443_set_feedback_src()***ｺﾏﾝﾄによって「外部のｴﾝｺｰﾀﾞｨｰﾄﾞﾊﾞｯｸ(EA、EB ﾋﾟﾝ)」信号を有効に設定しなければなりません。そして、ｺﾏﾝﾄﾞﾊﾟﾙｽと外部のﾌｨｰﾄﾞﾊﾞｯｸﾊﾟﾙｽ間の比率を***_7443_set_move_ratio()***ｺﾏﾝﾄﾞによって適切に設定されなければなりません。
_7443_start_tr_move()ｺﾏﾝﾄﾞでは、ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでの移動位置(距離)をﾊﾟﾙｽの単位で与えなければなりません。***_7443_start_ta_move()***ｺﾏﾝﾄﾞと***_7443_start_tr_move()***ｺﾏﾝﾄﾞは非対称型台形動作(加速 Tacc と減速 Tdec が同じでない)ですので加速と減速別々の値を指定することができます。***StrVel*** と ***MaxVel*** のﾊﾟﾗﾒｰﾀは１秒あたりのﾊﾟﾙｽ数(pps)で与えられます。***Tacc*** と ***Tdec*** のﾊﾟﾗﾒｰﾀは加速/減速の時間をそれぞれ秒で表します。
下記の公式は、１回の動作における物理的な意味(移動速度、加速時間、減速時間)がどのように計算されているかの基本的な関係式です。

```
MaxVel = StrVel + accel * Tacc;
StrVel = MaxVel + decel * Tdec;
```

accel/decel は pps/秒単位の加速/減速率を表します。台形動作の中の領域は移動距離を表します。

速度設定の単位は１秒あたりのﾊﾟﾙｽ数(pps)です。通常、ﾓｰﾀ、ﾄﾞﾗｲﾊﾞの取扱説明書では、単位速度は１分あたりの回転数(rpm)で表しています。単純な換算が、これらの２つの単位を合わせるのに必要です。ここで変換を説明するのに例を使用します。

例:

AB相のｴﾝｺｰﾀﾞがあるｻｰﾎﾞﾓｰﾀがX-Yﾃｰﾌﾞﾙに使用されています。ｴﾝｺｰﾀﾞの分解能は１回転あたり2000ﾊﾟﾙｽです。ﾓｰﾀの最高回転速度が3600rpmになっているとき、PPCI7443に設定しなければならない最大の出力周波数はどのくらいであるか?

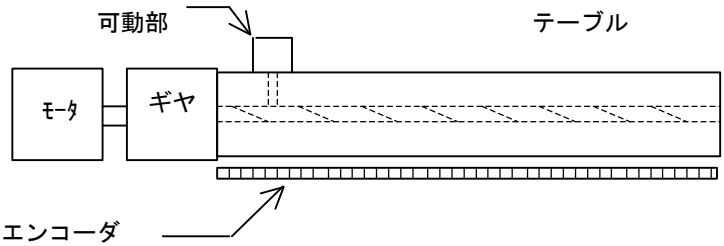
答え:

```
max_vel = 3600 / 60 * 2000 * 4
        = 48000pps
```

なぜ*4 がある理由は、AB 相毎に４つの状態があるからです。(ｾｸｼｮﾝ 4.4 の図参照)

機械的分解能がｺﾏﾝﾄﾞﾊﾟﾙｽの分解能と違っている場合、動作比率を設定する必要があります。例えば、ｲﾝｸﾘﾒﾝﾀﾙｴﾝｺｰﾀﾞが可動部の実際の位置を測定するために動くﾃｰﾌﾞﾙに取り付けられているとします。ｻｰﾎﾞﾓｰﾀは、ギヤ機構によって可動部を動かすのに使用されます。ギヤ機構は、ﾓｰﾀの回転する動きを線の動きに変換するのに使用されます(次の図参照)。ﾓｰﾀの分解能が１回転 8000 ﾊﾟﾙｽでギヤ機構の分解能が１回転 100mm とします(ﾓｰﾀが１回転すると可動部が 100mm 移動する)。このときのｺﾏﾝﾄﾞﾊﾟﾙｽの分解能は 80 ﾊﾟﾙｽ/mm です。テーブルにおけるｴﾝｺｰﾀﾞの取り付けの分解能は 200 ﾊﾟﾙｽ/mm です。それによりﾕｰｻﾞｰは機能の動作比率を 200/80=2.5 と設定する必要があります。

_7443_set_move_ratio(axis,2.5);



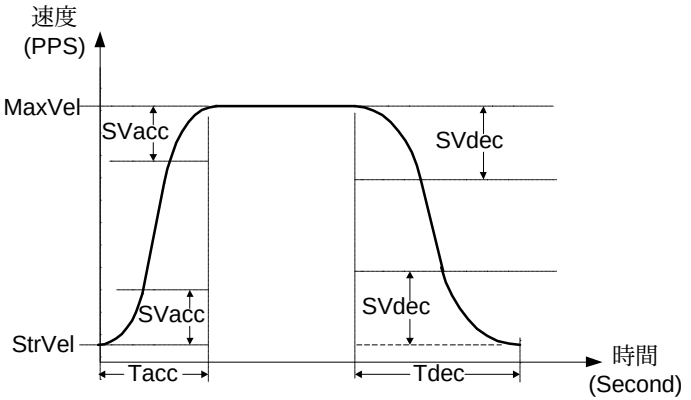
もし、動作開始ｺﾏﾝﾄﾞを発行する前にこの比率が設定されていなければ、「ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞ」で動作させるときに問題が発生します。なぜなら、PPCI7443 は動作中に実際のｱﾌﾞｿﾘｭｰﾄ位置を認識することができないからです。

◆ 関連ﾌｧﾝｸｼｮﾝ:
_7443_start_ta_move(),_7443_start_tr_move() : セクション6.6参照
_7443_motion_done() : セクション6.11参照
_7443_set_feedback_src() : セクション6.4参照
_7443_set_move_ratio() : セクション6.6参照

4.1.4 S字曲線動作

このﾓｰﾄﾞは、１個の軸(ﾓｰﾀ)をＳ字曲線動作で指定された位置(距離)まで動作させるのに使用されます。Ｓ字曲線動作はｽﾃｯﾋﾟﾝｸﾞﾓｰﾀとｻｰﾎﾞﾓｰﾀの両方に有効です。加速斜面の始まりと定速度への変化への滑らかな変遷は台形動作より少ない消耗となります。より滑らかな動作性はﾓｰﾀとｼｽﾃﾑ機能の寿命を増加させます。
Ｓ字曲線を動かすにはいくつかのﾊﾟﾗﾒｰﾀの設定が必要です。それらは下記の通りです。

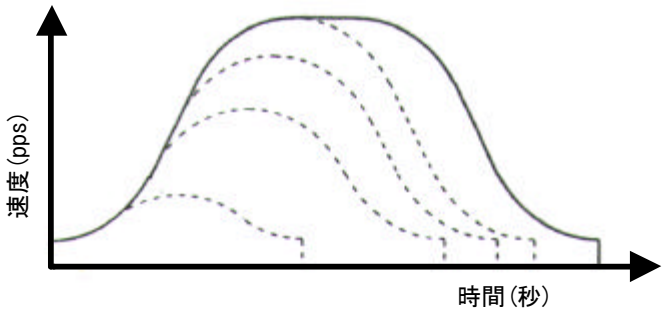
Pos:	ｱﾌﾞｿﾘｭｰﾄによる位置
Dist:	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞによる移動距離
StrVel:	移動開始速度
MaxVel:	移動最高速度
Tacc :	低速から高速への加速時間 (sec)
Tdec:	高速から低速への減速時間 (sec)
SVacc:	加速時間のＳ字曲線領域 (pps)
SVdec:	減速時間のＳ字曲線領域 (pps)



通常、加速/減速には１個の直線部分(linear)と２個のＳ字部分(SVacc/SVdec)の計３個の領域から成ります。Ｓ字部分の加速/減速(SVacc/SVdec)の間では、加速/減速が一定に変化し、そして直線部分(linear)の加速/減速が一定です。最初のＳ字部分の加速の間に速度がStrVel から(StrVel + SVacc)まで加速します。そして、２番目のＳ字部分の加速の間に速度が(MaxVel－SVacc)から MaxVel まで加速します。それらの間の直線部分の加速で、(StrVel + SVacc)から(MaxVel－SVacc)まで速度を一定に加速します。減速も同様です。

特別なケース:
もし直線部分の領域をなくしたい場合、SVacc/SVdec は「0」を設定して下さい。

SVacc/SVdec は pps の単位で、0～(MaxVel-Strvel)/2 の範囲で設定します。「0」を設定した場合は直線領域がないことを意味します。
Ｓ字曲線動作機能は、目標位置が直線部/Ｓ字曲線部の加速ﾊﾟﾗﾒｰﾀによって最高速度まで達しない距離(移動距離が MaxVel に達することができない短い距離)の場合、最高速度は自動的に下げらます。下記の図をご参照下さい。
規則は自動的に MaxVel と Tacc、Tdec、SVacc、SVdec の値を下げて、StrVel、加速、および急激な動作を変わりなく保つことです。台形動作についても同様に適応されます。



- 関連ﾌｧﾝｸｼｮﾝ:
  _7443_start_sr_move(),_7443_start_sa_move() : セクション6.6参照
  _7443_motion_done() : セクション6.11参照
  _7443_set_feedback_src() : セクション6.4参照
  _7443_set_move_ratio() : セクション6.6参照

下記の表は、ﾌﾟﾘｾｯﾄﾓｰﾄﾞ（台形動作とＳ字曲線動作の両方）と一定速度ﾓｰﾄﾞを含んだすべての１軸動作機能間の相違を表しています。

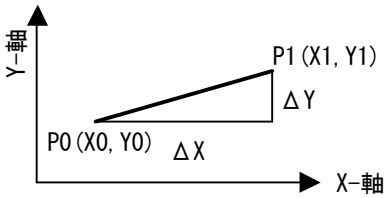
	速度動作			
	台形動作	Ｓ字動作	ｲﾝｸﾘﾒﾝﾀﾙ	ｱﾌﾞｿﾘｭｰﾄ
_7443_tv_move	○	×	----------	----------
_7443_sv_move	×	○	----------	----------
_7443_v_change	○	○	----------	----------
_7443_sd_stop	○	○	----------	----------
_7443_emg_stop()	----------	----------	----------	----------
_7443_start_ta_move	○	×	×	○
_7443_start_tr_move	○	×	○	×
_7443_start_sr_move	×	○	○	×
_7443_start_sa_move	×	○	×	○

4.1.5　２～４軸の直線補間

このﾓｰﾄﾞでは、４軸のうちの任意の２軸、３軸、４軸を選択し、直線補間動作を実行することができます。「多軸間の補間」とは、これらの軸が同時に移動開始し、同時に目標地点に到達することを意味します。直線補間とは、移動するすべての軸の速度の比率が一定の値であることを意味します。

◆　２軸直線補間

下記の図は、P0 から P1 まで XY（４軸のうち任意の２軸）を動かす直線補間です。２軸は同時に移動開始し、同時に停止します。そして、移動方向は直線です。



Ｘ軸とＹ軸に沿った速度比はそれぞれ（ΔX: ΔY）で、ﾍﾞｸﾄﾙ速度は下記の通りです。

$$\frac{\Delta P}{\Delta t} = \sqrt{(\frac{\Delta X}{\Delta t})^2 + (\frac{\Delta Y}{\Delta t})^2}$$

２軸の直線補間機能を呼び出すとき、ｽﾀｰﾄ速度(StrVel)と最高速度(MaxVel)はﾍﾞｸﾄﾙ速度を定義します。台形動作Ｓ字曲線動作両方とも利用は可能です。
例:

_7443_start_tr_move_xy(0,30000.0,40000.0,1000.0,5000.0,0.1,0.2)

この例は、基板番号０のＸＹ（１軸目と２軸目）の２軸直線補間動作を実行します。

ΔX = 30000 ﾊﾟﾙｽ
ΔY = 40000 ﾊﾟﾙｽ
ｽﾀｰﾄ速度
ﾍﾞｸﾄﾙ速度 = 1000pps, X軸速度 = 600pps, Y軸速度 = 800pps
最高速度
ﾍﾞｸﾄﾙ速度 = 5000pps, X軸速度 = 3000pps, Y軸速度 = 4000pps
加速時間 = 0.1秒
減速時間 = 0.2秒

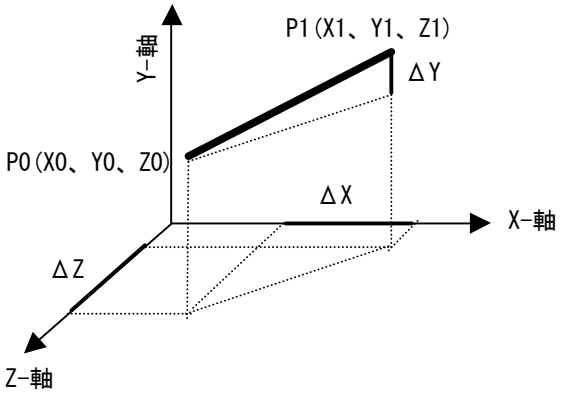
２軸直線補間には軸を割り当てるの機能に２つのｸﾞﾙｰﾌﾟがあります。最初のグループは４軸のうちXY(軸0と軸1)とそしてZU(軸2と軸3)に分けられます。これらのﾌｧﾝｸｼｮﾝを呼び出すことによって、移動軸は既に割り当てられます。
_7443_start_tr_move_xy(),_7443_start_tr_move_zu(),
_7443_start_ta_move_xy(),_7443_start_ta_move_zu(),
_7443_start_sr_move_xy(),_7443_start_sr_move_zu(),
_7443_start_sa_move_xy(),_7443_start_sa_move_zu(),
：セクション6.7参照

２番目のｸﾞﾙｰﾌﾟはﾕｰｻﾞｰが任意に２軸を割り当てる方法です。
_7443_start_tr_line2(),_7443_start_sr_line2(),
_7443_start_ta_line2(),_7443_start_sa_line2(),
：セクション6.7参照

_7443_startの後のｷｬﾗｸﾀ「t」,「s」,「r」,「a」の意味は
t： 台形動作
s： Ｓ字曲線動作
r： ｲﾝｸﾘﾒﾝﾀﾙ動作
a： ｱﾌﾞｿﾘｭｰﾄ動作

◆ ３軸直線補間

PPCI7443 の４軸のうち任意の３軸の直線補間を実行できます。下記の図は、P0 から P1 まで XYZ(軸 0、1、2 が選択されて、X、Y、Z にに割り当てられている場合)を動かす直線補間です。３軸は同時に移動開始し、同時に停止します。そして、移動方向は領域の直線です。



Ｘ軸、Ｙ軸、およびＺ軸に沿った速度比はそれぞれ(ΔX:ΔY:ΔZ)で、そして、ﾍﾞｸﾄﾙ速度は下記の通りです。

$$\frac{\Delta P}{\Delta t} = \sqrt{(\frac{\Delta X}{\Delta t})^2 + (\frac{\Delta Y}{\Delta t})^2 + (\frac{\Delta Z}{\Delta t})^2}$$

３軸の直線補間機能を呼び出すとき、ｽﾀｰﾄ速度(StrVel)と最高速度(MaxVel)はﾍﾞｸﾄﾙ速度を定義します。台形動作Ｓ字曲線動作両方とも利用は可能です。

例：

```
_7443_start_tr_line3(0,1000.0 /*ΔX */,2000.0/*ΔY */,3000.0
/*DistZ*/,100.0/*StrVel*/,5000.0/*MaxVel*/,0.1/*sec*/,0.2/*sec*/)
```

ΔX = 1000 ﾊﾟﾙｽ
ΔY = 2000 ﾊﾟﾙｽ
ΔZ = 3000 ﾊﾟﾙｽ

ｽﾀｰﾄ速度

ﾍﾞｸﾄﾙ速度 = 100pps,　　Ｘ軸速度 = $100 \div \sqrt{14}$ = 26.7pps
Ｙ軸速度 = $2 \times 100 \div \sqrt{14}$ = 53.3pps
Ｚ軸速度 = $3 \times 100 \div \sqrt{14}$ = 80.1pps

最高速度

ﾍﾞｸﾄﾙ速度 = 5000pps,　　Ｘ軸速度 = 5000÷$\sqrt{14}$ = 1336pps

Ｙ軸速度 = 2×5000÷$\sqrt{14}$ = 2672pps

Ｚ軸速度 = 3×5000÷$\sqrt{14}$ = 4008pps

３軸の直線補間と関連するﾌｧﾝｸｼｮﾝを下記に記述します。

_7443_start_tr_line3(),_7443_start_sr_line3(),
_7443_start_ta_line3(),_7443_start_sa_line3(),

：セクション6.7参照

_7443_start の後のｷｬﾗｸﾀ「t」,「s」,「r」,「a」の意味は

t： 台形動作
s： Ｓ字曲線動作
r： ｲﾝｸﾘﾒﾝﾀﾙ動作
a： ｱﾌﾞｿﾘｭｰﾄ動作

◆ ４軸直線補間

４軸の直線補間では、Ｘ軸、Ｙ軸、Ｚ軸、およびＵ軸に沿った速度比率はそれぞれ(ΔX:ΔY:ΔZ:ΔU)で、そしてﾍﾞｸﾄﾙ速度は下記の通りです。

$$\frac{\Delta P}{\Delta t}=\sqrt{(\frac{\Delta X}{\Delta t})^2+(\frac{\Delta Y}{\Delta t})^2+(\frac{\Delta Z}{\Delta t})^2+(\frac{\Delta U}{\Delta t})^2}$$

４軸の直線補間と関連するﾌｧﾝｸｼｮﾝを下記に記述します。

_7443_start_tr_line4(),_7443_start_sr_line4(),
_7443_start_ta_line4(),_7443_start_sa_line4(),

：セクション6.7参照

_7443_start の後のｷｬﾗｸﾀ「t」,「s」,「r」,「a」の意味は

t： 台形動作
s： Ｓ字曲線動作
r： ｲﾝｸﾘﾒﾝﾀﾙ動作
a： ｱﾌﾞｿﾘｭｰﾄ動作

◆ 連続直線補間ﾓｰﾄﾞ

２から４軸の直線補間には、連続補間ﾓｰﾄﾞがあります。このﾓｰﾄﾞを連続に設定すると停止位置が距離ではなく、停止ｺﾏﾝﾄﾞが書かれるまで補間動作を継続します。

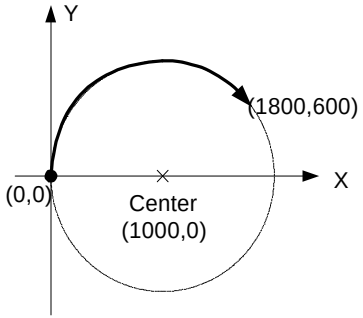
_7443_set_line_mode()ｺﾏﾝﾄﾞの mode ﾊﾟﾗﾒｰﾀを「1」に設定すると連続補間ﾓｰﾄﾞになり、「0」を設定すると位置決め補間ﾓｰﾄﾞになります。

_7443_set_line_mode() : セクション 6.7 参照

4.1.6 ２軸円弧補間

PPCI7443 の４軸のうち任意の２軸の円弧補間ができます。例は、X,Y 軸の位置(0,0)から、位置(1800,600)までの円弧補間です。その移動間の軌道は円弧で、最高速度(MaxVel)はﾀﾝｼﾞｪﾝﾄ速度です。

例：
_7443_start_arc_xy(0/*card No*/,1000.0/*center X*/,0/*center Y*/,1800.0/*END X*/,600.0 /*End Y*/1000.0/*MaxVel*/)



円弧補間の軌道を指定するために、下記のﾊﾟﾗﾒｰﾀは明確に定義されなければなりません。

Center Point :	円弧の中心座標 (ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞ) or 円弧の中心へのｵﾌｾｯﾄ距離 (ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞ)
End Point :	円弧の終了座標 (ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞ) or 円弧の中心へのｵﾌｾｯﾄ距離 (ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞ)
Direction :	移動方向 (CW or CCW)

上の情報で十分制限できるので、円弧の半径か角度を設定する必要はありません。２軸のどちらかが終了位置に到達したとき、円弧動作は停止します。

２軸円弧補間には軸を割り付ける機能として２つのｸﾞﾙｰﾌﾟがあります。最初のグループは４軸のうち XY(軸 0 と軸 1)と ZU(軸 2 と軸 3)に割り当てます。これらのﾌｧﾝｸｼｮﾝを呼び出すことで移動させる軸はあらかじめ割り当てられています。

_7443_start_r_arc_xy(),_7443_start_r_arc_zu(),
_7443_start_a_arc_xy(),_7443_start_a_arc_zu(),
_7443_start_tr_arc_xy(),_7443_start_tr_arc_zu(),
_7443_start_ta_arc_xy(),_7443_start_ta_arc_zu(),
_7443_start_sr_arc_xy(),_7443_start_sr_arc_zu(),
_7443_start_sa_arc_xy(),_7443_start_sa_arc_zu()
：セクション6.8参照

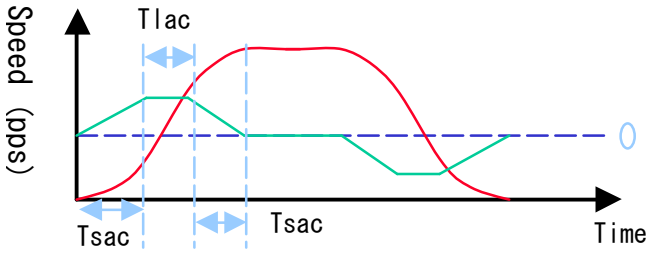
_7443_start の後のｷｬﾗｸﾀ「t」,「s」,「r」,「a」の意味は
t： 台形動作
s： Ｓ字曲線動作
r： ｲﾝｸﾘﾒﾝﾀﾙ動作
a： ｱﾌﾞｿﾘｭｰﾄ動作
を意味します。「t」及び「s」がない円弧補間は加減速がない円弧補間です。

２番目のｸﾞﾙｰﾌﾟはﾕｰｻﾞｰが任意に２軸を割り当てる方法です。
_7443_start_r_arc2(),_7443_start_a_arc2(),
_7443_start_tr_arc2(),_7443_start_ta_arc2(),
_7443_start_sr_arc2(),_7443_start_sa_arc2()
：セクション6.8参照

_7443_start の後のｷｬﾗｸﾀ「t」,「s」,「r」,「a」の意味は
t： 台形動作
s： Ｓ字曲線動作
r： ｲﾝｸﾘﾒﾝﾀﾙ動作
a： ｱﾌﾞｿﾘｭｰﾄ動作
を意味します。「t」及び「s」がない円弧補間は加減速がない円弧補間です。

4.1.7　　加速/減速機能付き円弧補間

ｾｸｼｮﾝ 4.1.6 では、加速/減速の機能のない円弧補間動作の説明です。すなわち、台形ｶｰﾌﾞまたはＳ字ｶｰﾌﾞによる加減速が行えない事です。時には機械を滑らかに動作させるために円弧補間動作でも加減速を必要とすることがあります。PPCI7443 にはその動作を行うためにもう一つの円弧補間のｸﾞﾙｰﾌﾟがあります。例えば軸０と軸１で台形加減速による円弧補間を行うとします。その動作は
_7443_start_tr_arc_xy()ｺﾏﾝﾄﾞにより実行できます。このｺﾏﾝﾄﾞ名は軸０と軸１を台形加減速で円弧補間動作させる事です。これらの機能の全リストについてはｾｸｼｮﾝ 6.8 を参照して下さい。
加減速付きの円弧補間動作は減速ポイントを自動設定にして使用して下さい。自動設定の場合、加速と減速に設定する値は
（減速時間）　≦　（加速時間×２）の範囲で設定して下さい。
減速ポイントを自動設定にするコマンドは***_7443_set_auto_rdp()***です。
詳しくはｾｸｼｮﾝ 6.6 を参照して下さい。

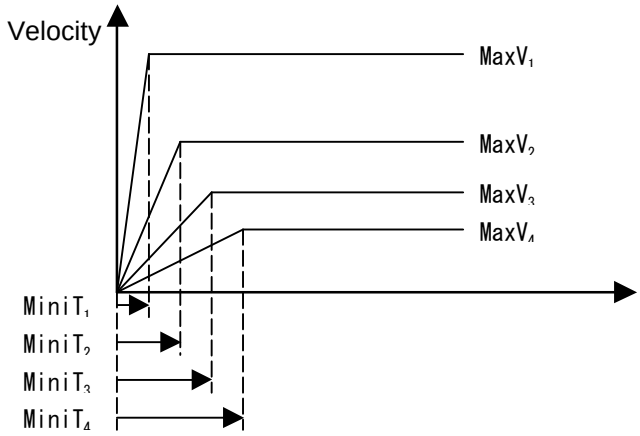


4.1.8　速度と加速時間の関係

軸動作機能の最高速度ﾊﾟﾗﾒｰﾀに達するまでの加速時間には、最小設定値があります。すなわち、一つの動作において最大値と最小値の範囲があることを意味します。時には、加速時間の最小設定値を得る事を必要とします。もしそれを望むならば、その最小加速時間に一致する最高速度を設定する必要があります。そのためのｺﾏﾝﾄﾞとして ***_7443_fix_speed_range()***ｺﾏﾝﾄﾞがあります。この機能は短い加速時間で最高速度まで上げることができます。けれどもそれは実際の速度に影響を及ぼさないでしょう。例えば、速度を 0pps から 5000pps まで 1msec で加速する場合、この設定ではできません。軸動作の前により速い速度の設定をすることで行うことができます。ﾌﾟﾛｸﾞﾗﾑは以下のようになります。

```
_7443_fix_speed_range(AxisNo,OverVelocity);

_7443_start_tr_move(AxisNo,5000,0,5000,0.001,0.001);
```



上記の_7443_fix_speed_range()ｺﾏﾝﾄﾞ内の「OverVelocity」の最適値を決定する方法として、***_7443_verify_speed()***ｺﾏﾝﾄﾞがあります。このｺﾏﾝﾄﾞの入力値は、軸動作の開始速度(StrVel)、最高速度(MaxVel)、そしてｵｰﾊﾞｰｽﾋﾟｰﾄﾞです。出力値は加速時間の最小値と最大値です。
例えば、以下のｺﾏﾝﾄﾞの加速時間の設定範囲を調べてみます。

```
_7443_start_tr_move (AxisNo,5000,0,5000,0.001,0.001);
```

以下のｺﾏﾝﾄﾞを実行します。

```
_7443_verify_speed (0,5000,&minAccT,&maxAccT,5000);
```

結果、minAccT は 0.0267sec で maxAccT は 873.587sec となりました。こ

の最小設定時間では 1msec での加速はできませんので、
_7443_fix_speed_range()ｺﾏﾝﾄﾞにより最高速度を上げる必要があります。
もし、設定速度を「20000」に設定したならば、

```
_7443_verify_speed (0,5000,&minAccT,&maxAccT,20000);
```

minAccT は 0.00666sec で maxAccT は 218.387sec となります。しかしまだ最小設定値には足りません。
今度は設定速度を「140000」に設定したならば、

```
_7443_verify_speed (0,5000,&minAccT,&maxAccT,140000);
```

minAccT は 0.000948sec で maxAccT は 31.08sec となります。この最小加速時間は十分な値です。そこでﾌﾟﾛｸﾞﾗﾑは以下のようになります。

```
_7443_fix_speed_range(AxisNo,140000);

_7443_start_tr_move(AxisNo,5000,0,5000,0.001,0.001);
```

注意１

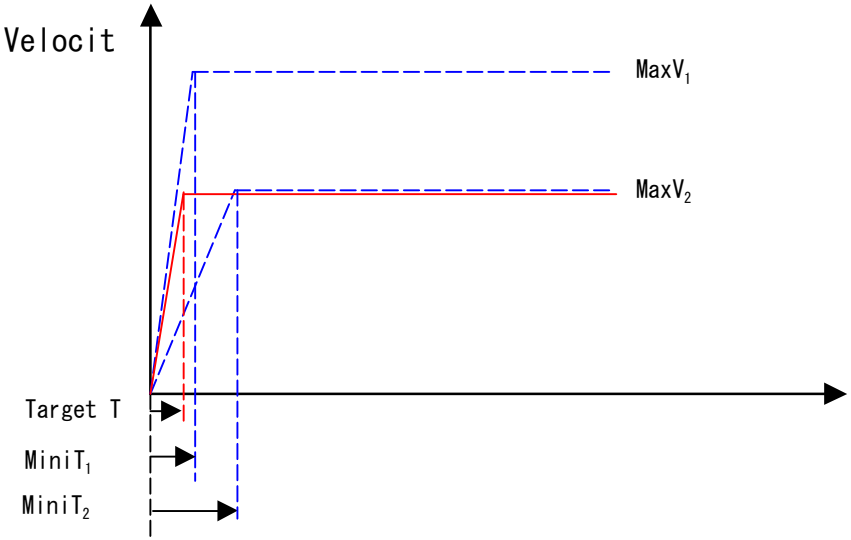
_7443_verify_speed()ｺﾏﾝﾄﾞの戻り値は動作させる軸ｺﾏﾝﾄﾞの最小速度であって開始速度値と等しいわけではありません。上記の例で開始速度の設定値は 0pps ですが実際には 3pps となります。

注意２

_7443_fix_speed_range()ｺﾏﾝﾄﾞによって設定された最高速度が不要な場合、***_7443_unfix_speed_range()***ｺﾏﾝﾄﾞにより解除することができます。

注意３

常にｵｰﾊﾞｰｽﾋﾟｰﾄﾞを使おうとしないで、いつ必要かを知っておかなければなりません。ｵｰﾊﾞｰｽﾋﾟｰﾄﾞをより大きくすることにより、速度の分解能が粗くなるの覚えていてください。



例：

希望値 ：(MaxV$_2$, Target T)

しかし、(MaxV, MiniT) の関係により MaxV$_2$ への加速時間は希望値より長くなります。そこで(MaxV, MiniT) の関係をより高い物に(MaxV$_1$, MiniT$_1$) に変更しなければなりません。最終的にｺﾏﾝﾄﾞは以下のようになります。

```
_7443_fix_speed_range(AxisNo, MaxV1);

_7443_start_tr_move(AxisNo,Distance,0, MaxV2, TargetT, Target T);
```

- 関連ﾌｧﾝｸｼｮﾝ

  _7443_fix_speed_range(), _7443_unfix_speed_range(), _7443_verify_speed()

  ：セクション6.5参照

4.1.9　連続動作

PPCI7443 は連続した動作を実行することが可能です。１軸の動作ﾓｰﾄﾞ（台形動作：ｾｸｼｮﾝ 4.1.3 参照，Ｓ字曲線動作：ｾｸｼｮﾝ 4.1.4 参照）と複数軸の補間動作（直線補間：4.1.5 参照，円弧補間：4.1.6 参照）の両方を連続に行うことができます。

例えば、１軸のﾌﾟﾘｾｯﾄ動作を行うのに次のﾌｧﾝｸｼｮﾝを呼び出します。

_7443_start_ta_move(0,50000.0,100.0,30000.0,0.1,0.0)

軸０を「50000.0」の位置に移動させ、軸が目標位置に到達する前に、２番目のﾌﾟﾘｾｯﾄ動作を呼び出すことができます。

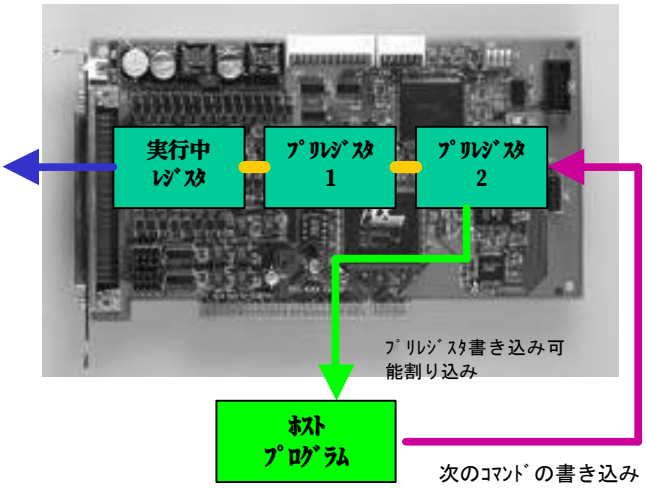
_7443_start_tr_move(0,20000.0,100.0,30000.0,0.0,0.2)

２番目のﾌｧﾝｸｼｮﾝ呼び出しは最初のｺﾏﾝﾄﾞの実行には影響しないで、それは PPCI7443 のﾌﾟﾘﾚｼﾞｽﾀに書かれます。最初のｺﾏﾝﾄﾞが完了した後、ﾌﾟﾘﾚｼﾞｽﾀ値に従って、PPCI7443 は２番目の動作を続けます。それにより、この２つの動作の間には時間間隔がなくなります。２番目の出力ﾊﾟﾙｽは１番目の動作が位置「50000.0」に到達した瞬間に発生します。

連続した動作の理論を下記に説明します。

◆　連続動作の理論

下記の図は PPCI7443 のﾚｼﾞｽﾀﾃﾞｰﾀの流れを示します。



ｽﾃｯﾌﾟ 0: すべてのﾚｼﾞｽﾀとﾌﾟﾘﾚｼﾞｽﾀは空です。
ｽﾃｯﾌﾟ 1: 最初の動作が実行され、CPU がﾌﾟﾘﾚｼﾞｽﾀ 2 に対応する値を書き込みます。
_7443_start_ta_move(0,50000.0,100.0,30000.0,0.1,0.0)
ｽﾃｯﾌﾟ 2: ﾌﾟﾘﾚｼﾞｽﾀ 1 とﾚｼﾞｽﾀが空であるので、ﾌﾟﾘﾚｼﾞｽﾀ 2 のﾃﾞｰﾀは自動的にﾚｼﾞｽﾀに移動され、そして PCL6045 により実行されます。
ｽﾃｯﾌﾟ 3: ２番目のﾌｧﾝｸｼｮﾝが呼ばれ、CPU がﾌﾟﾘﾚｼﾞｽﾀ 2 に対応する値を書き込みます。
_7443_start_tr_move(0,20000.0,100.0,30000.0,0.0,0.2)
ｽﾃｯﾌﾟ 4: ﾌﾟﾘﾚｼﾞｽﾀ 1 が空であるので、ﾌﾟﾘﾚｼﾞｽﾀ 2 のﾃﾞｰﾀは自動的にﾌﾟﾘﾚｼﾞｽﾀ 1 に移動して、実行を待ちます。
ｽﾃｯﾌﾟ 5: ３番目のﾌｧﾝｸｼｮﾝが呼ばれ、ﾌﾟﾘﾚｼﾞｽﾀ 2 に記憶されます。
ｽﾃｯﾌﾟ 6: 最初のﾌｧﾝｸｼｮﾝが終わるとﾚｼﾞｽﾀは空になり、ﾌﾟﾘﾚｼﾞｽﾀ 1 のﾃﾞｰﾀが PCL6045 によって即座にﾚｼﾞｽﾀに移動し実行されます。そして、ﾌﾟﾘﾚｼﾞｽﾀ 2 のﾃﾞｰﾀはﾌﾟﾘﾚｼﾞｽﾀ 1 に移動します。
ｽﾃｯﾌﾟ 7:PCL6045 により動作の終了を割り込み信号で CPU に知らせます。そして、ﾌﾟﾘﾚｼﾞｽﾀ 2 に４番目の動作を書き込むことができます。

◆ 連続動作を実行する手順

下記に連続動作を行う手順を記します。

ｽﾃｯﾌﾟ 1: (Dos 環境)
_7443_int_contol()による割り込み運転を可能にする。
(Windows 環境)
_7443_int_contol()と***_7443_int_enable()***による割り込み運転を可能にする。
ｽﾃｯﾌﾟ 2: ***_7443_set_int_factor()***で INT 要素のﾋﾞｯﾄ 2 を「True」に設定する。
ｽﾃｯﾌﾟ 3: ***_7443_set_continuous_move()***で ***conti_logic*** を「１」に設定する
(注意: このﾌｧﾝｸｼｮﾝはすべての動作がｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞであるならば、無視されます。)
ｽﾃｯﾌﾟ 4: 最初の３つの動作ﾌｧﾝｸｼｮﾝを呼び出します。
ｽﾃｯﾌﾟ 5: ﾌﾟﾘﾚｼﾞｽﾀ書き込み可能割り込みの EVENT(Windows 環境)か INT(Dos 環境)を待ちます。
ｽﾃｯﾌﾟ 6: ４番目の動作ﾌｧﾝｸｼｮﾝを呼び出します。
ｽﾃｯﾌﾟ 7: ﾌﾟﾘﾚｼﾞｽﾀ書き込み可能割り込みの EVENT(Windows 環境)か INT(Dos 環境)を待ちます。
ｽﾃｯﾌﾟ 8: ５番目の動作ﾌｧﾝｸｼｮﾝを呼び出します。

(ｽﾃｯﾌﾟ 7、8 繰り返し……続く…)
ｽﾃｯﾌﾟ n: 最後の動作ﾌｧﾝｸｼｮﾝを呼び出しそして、すべての完了を待ちます。
(注意:動作の完了を検出するもう一つの方法はﾎﾟｰﾘﾝｸﾞです。ﾕｰｻﾞｰは常に***_7443_check_continuous_buffer()***でバッファ状態をﾁｪｯｸします。)

◆ 連続動作の制限

下記の文は連続動作のための制限と提案です。

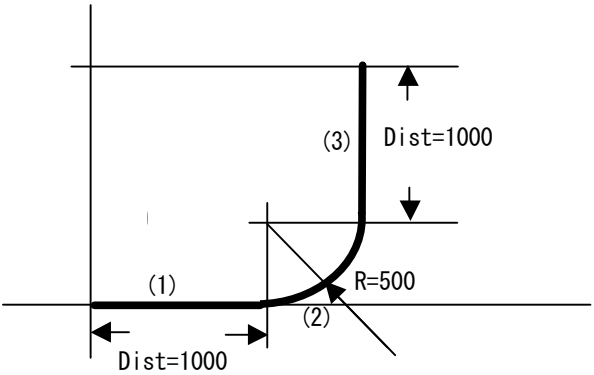
1. ﾌﾟﾘﾚｼﾞｽﾀが空でない間、ﾕｰｻﾞｰはそれ以上の動作を実行してはなりません。さもなければ、新しいｺﾏﾝﾄﾞはﾌﾟﾘﾚｼﾞｽﾀ 2 により上書きされます。
2. ２つの動作間の速度の連続を得るために、次の開始速度と前の終了速度は同じでなければなりません。これを達成するいくつかの方法があります。最も簡単な方法は減速/加速時間に「0」を設定することです。

例えば:

最初の動作： ***_7443_start_tr_move_XY(0,1000,0,0,5000,0.2, 0.0)***
(ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞによる２軸直線補間、Ｘ距離=1000、Ｙ距離=0、開始速度=0、最高速度=5000、Tacc=0.2、Tdec=0)

２番目の動作：***_7443_start_r_arc_xy(0,0,500,500,500,1,5000)***
(ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞによる２軸円弧補間、中心Ｘ=0、中心Ｙ=500、終了Ｘ=500、終了Ｙ=500、最高速度=5000、円弧は 4 分の 1 の CCW 円で、速度=5000)

３番目の動作：***_7443_start_tr_move_XY(0,0,1000,0,5000,0.0, 0.2)***
(ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞによる２軸直線補間、Ｘ距離=0、Ｙ距離=1000、開始速度=0、最高速度=5000、Tacc=0.0、Tdec=0.2)

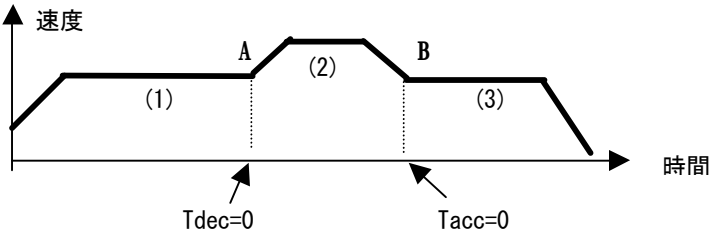


例の説明：
これらの３つの動作は動作間に時間待ちがなく、連続して実行されます。１番目はﾚｼﾞｽﾀに書き込まれ、即座に実行され、２番目はﾌﾟﾘﾚｼﾞｽﾀ１に書き込まれ、１番目の完了を待ちます。３番目はﾌﾟﾘﾚｼﾞｽﾀ２に書き込まれ、第２番目の完了を待ちます。 最初の動作の減速時間が「0」で完了となり、続いて２番目の動作の円弧補間が一定速度で動作します。
PPCI7443 はその１番目と２番目の交差する位置で、１番目の最高速度である一定の周波数を出力します。

1. それぞれの軸がﾚｼﾞｽﾀとﾌﾟﾘﾚｼﾞｽﾀを持つため、異なった軸間での連続動作は無関係です。
2. 連続動作では、異なった軸数の関数は許可できません。例えば、***_7443_start_ta_move_XY()***の後に ***_7443_start_tr_move()***を続けることはできません。逆もまた同様です。なぜならば、これら２つのﾌｧﾝｸｼｮﾝが個別に１軸ﾓｰﾄﾞと２軸ﾓｰﾄﾞであるからです。
3. ３軸か４軸の直線補間を連続して行うのは可能です。しかし、速度の継続を達成するのは不可能です。
4. もしｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞで連続した動作を行う場合は、少なくとも一度***_7443_reset_target_pos()***ｺﾏﾝﾄﾞで原点復帰を行って位置を明らかにしなければなりません。(4.1.8:原点復帰を参照)

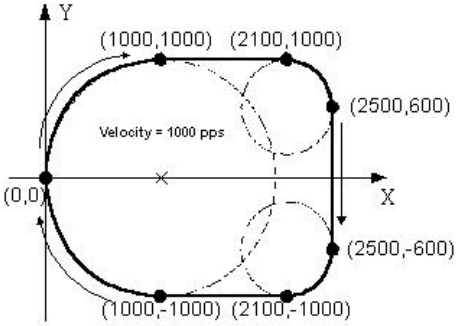
◆ 連続動作に関する例
下記は連続した動作に関する例です。

1. １軸の連続動作：あらかじめ設定された位置で速度を変える。



この例は事前に設定した位置で速度の変化を実行する連続した動作機能を使用する方法を示します。最初の動作(ta)は軸をＡ位置まで減速時間「0」で移動させ、その後、２番目の動作が直ちに継続します。(2)のｽﾀｰﾄ速度は(1)の最高速度と同じですので、速度の連続性がＡに存在しました。Ｂ位置において(3)の加速は「0」であるので速度の連続性が同じく築かれます。

2. 連続した２軸の補間：



この例は２軸の連続補間を実行する連続した動作機能を使用する方法を示します。このｱﾌﾟﾘｹｰｼｮﾝで速度の連続性は重要なﾎﾟｲﾝﾄです。前のﾍﾟｰｼﾞの例をご参照下さい。

- 関連ﾌｧﾝｸｼｮﾝ
  _7443_set_continuous_move(),_7443_check_continuous_buffer()
  ：セクション6.17参照

4.1.10　原点復帰モード

このﾓｰﾄﾞは***_7443_home_move()***ｺﾏﾝﾄﾞを書いた後、原点復帰が完了（条件を満たすまで）するまで PPCI7443 からﾊﾟﾙｽを出力させることができます（原点復帰完了でﾊﾟﾙｽ出力停止）。PPCI7443 には 13 種類の原点復帰ﾓｰﾄﾞがあります。***_7443_set_home_config()***ｺﾏﾝﾄﾞは ***home_mode*** を選択するのに使用されます。原点復帰移動開始方向は MaxVel に設定する符号によります。

原点復帰完了後、すべての位置関連の情報が「0」にﾘｾｯﾄされます。
PPCI7443 には、４個のｶｳﾝﾀと 1 個のｿﾌﾄｳｪｱで維持されたﾎﾟｼﾞｼｮﾝﾚｺｰﾀﾞがあります。それらは下記の通りです。

ｺﾏﾝﾄﾞｶｳﾝﾀ：	出力ﾊﾟﾙｽの数をｶｳﾝﾄする
ﾌｨｰﾄﾞﾊﾞｯｸｶｳﾝﾀ：	入力ﾊﾟﾙｽをｶｳﾝﾄする
ﾎﾟｼﾞｼｮﾝｴﾗｰｶｳﾝﾀ：	ｺﾏﾝﾄﾞﾊﾟﾙｽとﾌｨｰﾄﾞﾊﾞｯｸﾊﾟﾙｽの違いをｶｳﾝﾄ
汎用ｶｳﾝﾀ：	ﾊﾟﾙｽ出力、ﾌｨｰﾄﾞﾊﾞｯｸﾊﾟﾙｽ、手動ﾊﾟﾙｻｰまたは CLK/2 としてｶｳﾝﾀ入力を構成する
ﾀｰｹﾞｯﾄﾎﾟｼﾞｼｮﾝﾚｺｰﾀﾞ：	目標位置を記録する

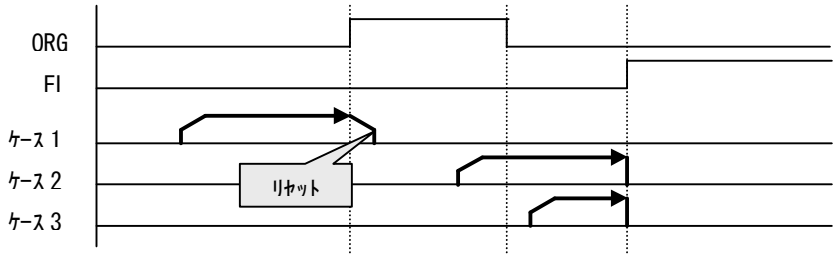
位置ｶｳﾝﾀに関するその他の詳細説明についてはｾｸｼｮﾝ 4.4 をご参照下さい。

原点復帰完了後、最初の４個のｶｳﾝﾀは、自動的に「0」にｸﾘｱされます。しかし、ﾀｰｹﾞｯﾄﾎﾟｼﾞｼｮﾝﾚｺｰﾀﾞはｸﾘｱしません。なぜなら、ｿﾌﾄｳｪｱによって保持されているからで、***_7443_reset_target_pos()***ｺﾏﾝﾄﾞを呼び出してﾀｰｹﾞｯﾄﾎﾟｼﾞｼｮﾝを「0」に設定する必要があります。それによりすべての位置情報は「0」になります。

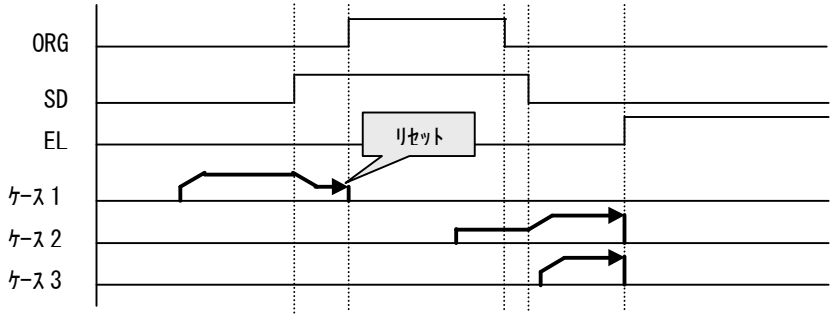
次ページから図は、いろいろな原点復帰ﾓｰﾄﾞとｶｳﾝﾀを「0」にｸﾘｱするﾎﾟｲﾝﾄを示します。

◆ home_mode=0:ORG → 減速 → 停止

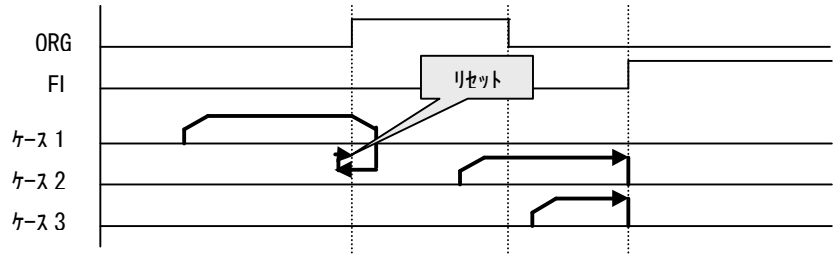
- SD(減速信号)が無効のとき



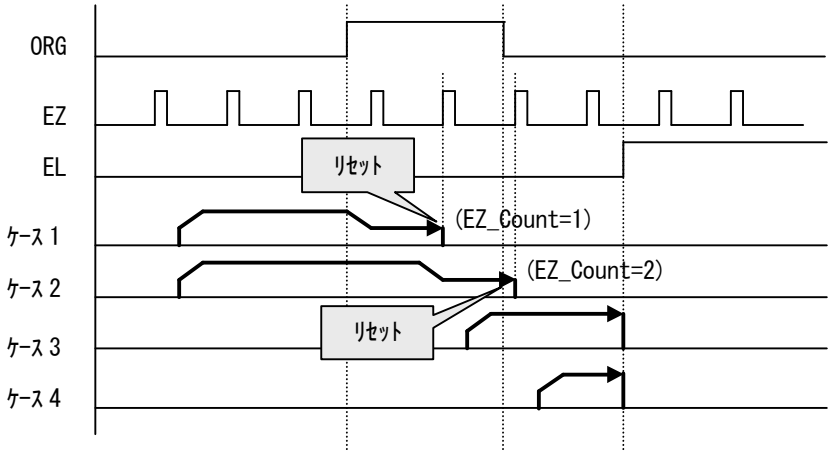
- SD(減速信号)が有効のとき



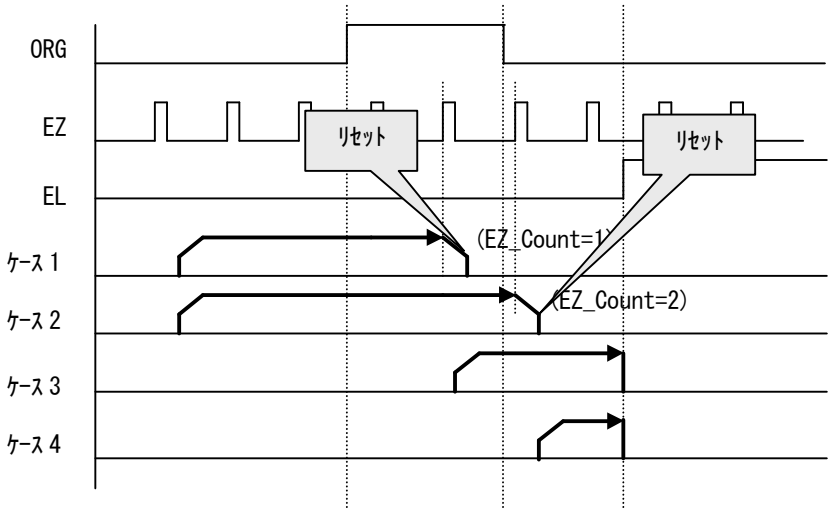
◆ home_mode=1:ORG → 減速 → ORG の端に停止



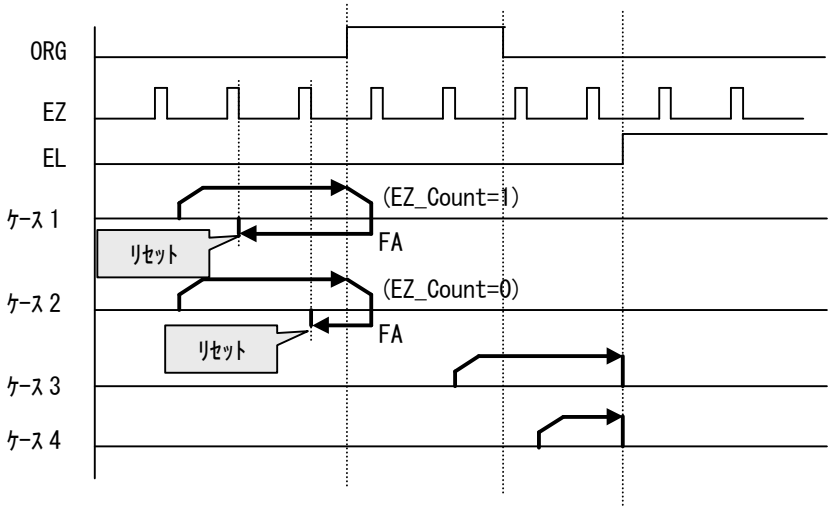
◆ home_mode=2:ORG → 減速 → EZ で停止



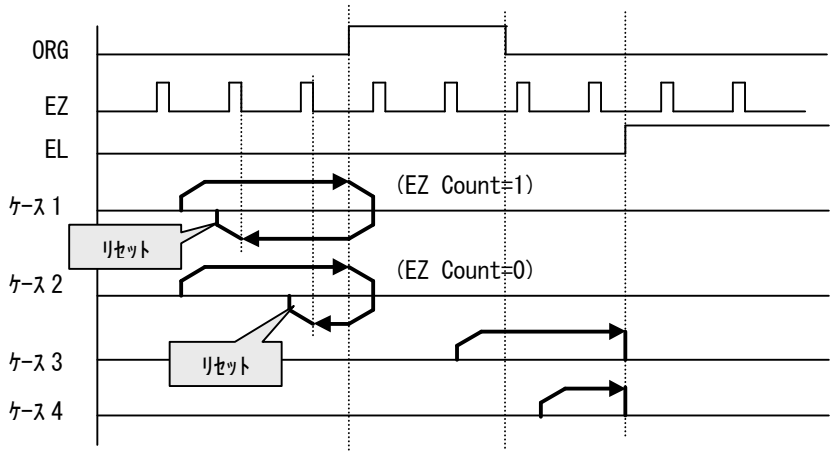
◆ home_mode=3:ORG → EZ → 減速 → 停止



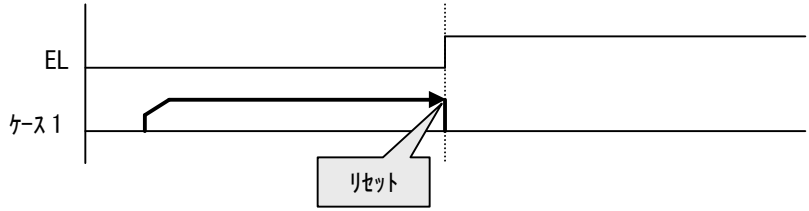
◆ home_mode=4:ORG → 減速 → FA 速度で戻る → EZ → 停止



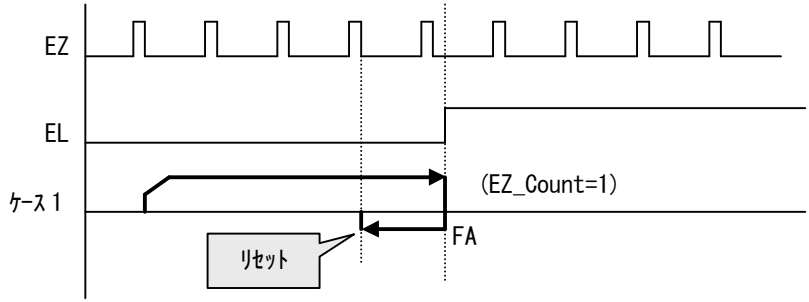
◆ home_mode=5:ORG → 減速 → 戻る → 高速へ加速 → EZ → 減速 → 停止



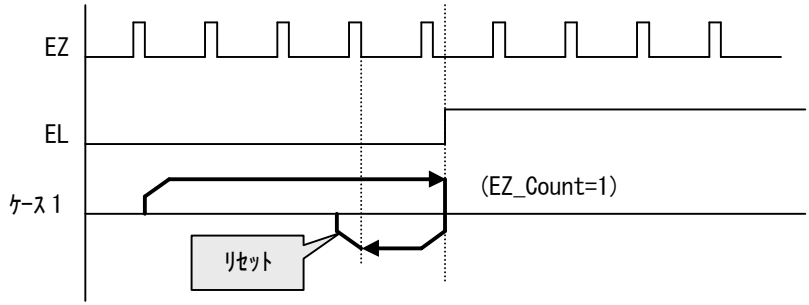
◆ home_mode=6:EL のみ



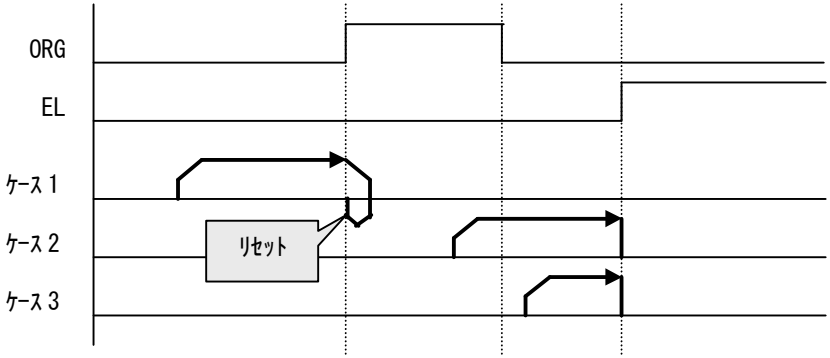
◆ home_mode=7:EL → 戻る → EZ で停止



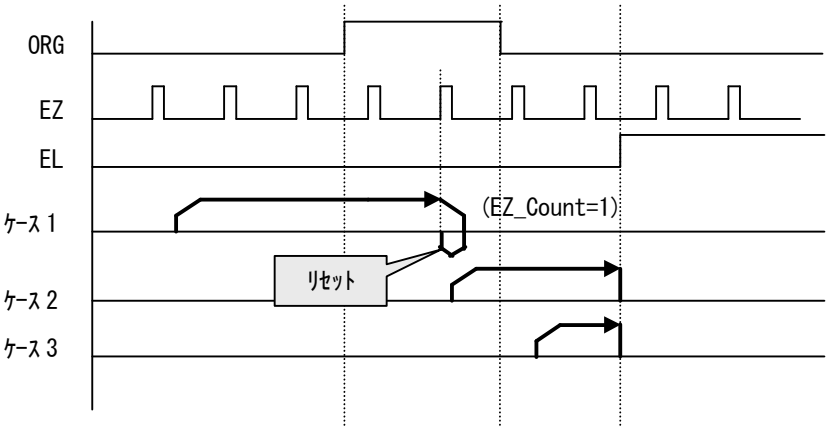
◆ home_mode=8: EL → 戻る → 高速へ加速 → EZ → 減速 → 停止



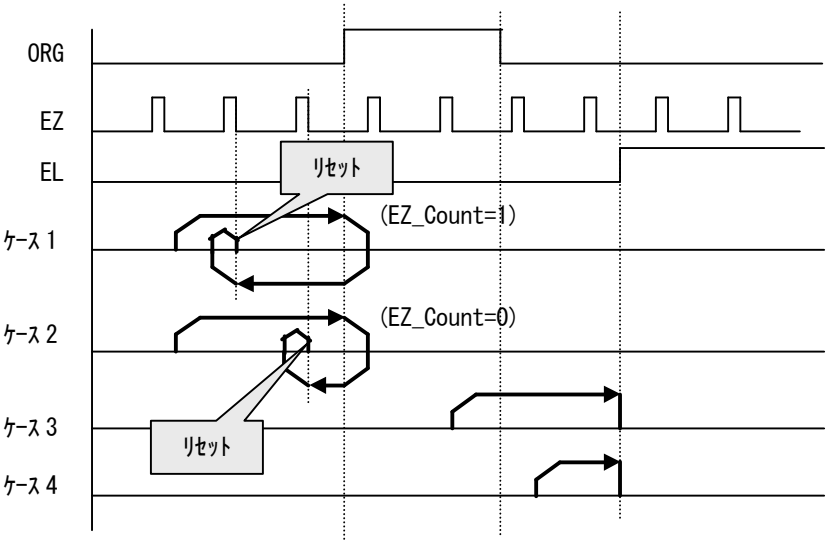
◆ home_mode=9:ORG → 減速 → 戻る → ORG の最初の端で停止



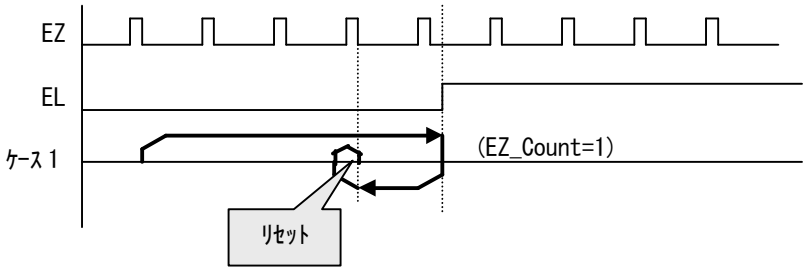
◆ home_mode=10:ORG → EZ → 減速 → 戻る → EZ の最初の端で停止



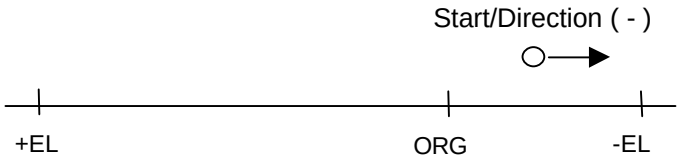
◆ home_mode=11:ORG → 減速 → 戻る(Back) → 高速へ加速 → EZ → 減速 → 戻る(Forward) → EZ の最初の端で停止



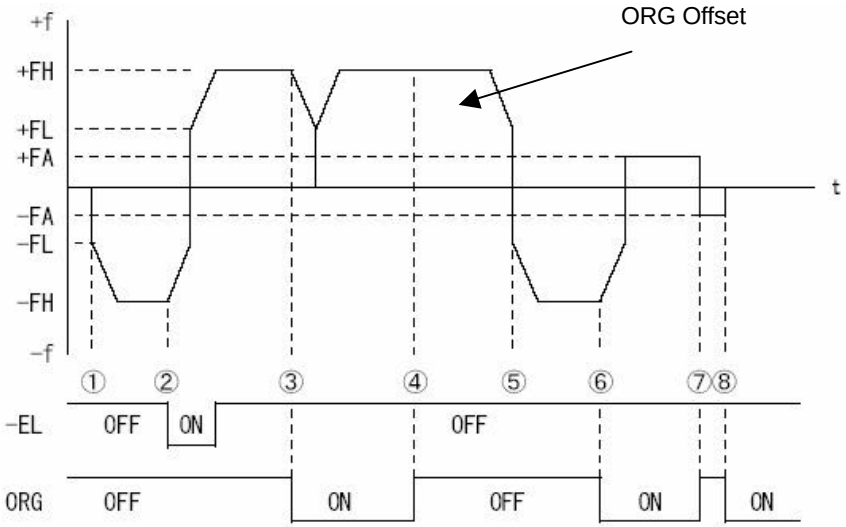
◆ home_mode=12:EL → 停止 → 戻る(Back) → 高速へ加速 → EZ → 減速 → 戻る(Forward) → EZ の最初の端で停止



◆ ホームサーチ機能例 (Home Mode = 1)



FL=Star Velocity
FH=Max Velocity (the sign stands for direction)
FA=Search Speed (half of the FH)



◆ 関連ﾌｧﾝｸｼｮﾝ:

_7443_set_home_config(), _7443_home_move(), _7443_home_search(), _7443_auto_home_seach() :

セクション6.9参照

4.1.11　手動ﾊﾟﾙｻｰﾓｰﾄﾞ

装置の手動操作のために、回転式のｴﾝｺｰﾀﾞような手動ﾊﾟﾙｻｰの使用が可能です。PPCI7443 はﾊﾟﾙｻｰからの入力信号を OUT と DIR ﾋﾟﾝから対応するﾊﾟﾙｽを出力することができます。それにより、外部の回路を簡素化して、軸の現在位置を制御することが可能になります。このﾓｰﾄﾞは、***_7443_pulser_vmove(), _7443_pulser_pmove()***または ***_7443_pulser_home_move()***ｺﾏﾝﾄﾞが呼ばれたとき、有効になります。***_7443_sd_stop()*** ｺﾏﾝﾄﾞか***_7443_emg_stop()***ｺﾏﾝﾄﾞもしくは動作の完了で停止します。

PPCI7443 はﾊﾟﾙｻｰから PA と PB ﾋﾟﾝに 90 度位相差の信号(AB 相)か＋ﾊﾟﾙｽかｰﾊﾟﾙｽ(CW/CCW)信号を入力します。ﾊﾟﾙｻｰの入力信号ﾓｰﾄﾞを設定するために、***_7443_set_pulser_iptmode()***ｺﾏﾝﾄﾞを使用します。90 度の位相差信号は１倍、２倍または４倍にして受信することもできます。もし、AB 相入力ﾓｰﾄﾞが選択されていれば、PA と PB 信号は 90 度の位相差の信号です。そして、PA 信号が PB 信号より 90 度進んでいるとき、ﾎﾟｼﾞｼｮﾝｶｳﾝﾀは増加します。

◆ 関連ﾌｧﾝｸｼｮﾝ:

_7443_pulser_vmove(),_7443_pulser_pmove(), _7443_pulser_home_move(),_7443_set_pulser_iptmode() : セクション6.10参照

4.1.12　ﾀｲﾏｰﾓｰﾄﾞ

この機能はプログラムを指定時間(msec)遅らせて実行するのに使用します。

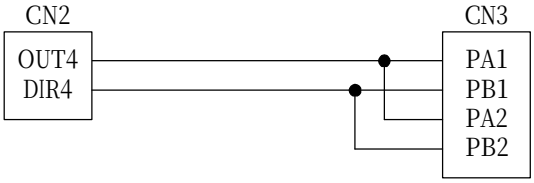
例. _7443_delay_time (0, 100);のｺﾏﾝﾄﾞを実行すると 100msec 遅れて次のプログラムが実行します。

◆ 関連ﾌｧﾝｸｼｮﾝ:

_7443_delay_time() : セクション6.1参照

4.1.13　ﾊﾟﾙｻｰによる補間

ﾊﾟﾙｻｰによる補間動作（任意２軸直線補間、任意２軸円弧補間）を行うことができます。この補間動作はｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでのみ動作できます。また下図のように、１軸をダミー軸として使用する場合は、その軸を含む補間動作はできません。下記の例では、４軸目をダミー軸とし、これにより１軸目から３軸目の内、任意の２軸による補間動作が行えます。



◆　任意の２軸直線補間

上記のように接続後、***_7443_pulser_r_line2()***ｺﾏﾝﾄﾞによりﾊﾟﾙｻｰによる２軸直線補間が行えます。ﾊﾟﾗﾒｰﾀ AxisArray により動作させる軸の選択ができます。

_7443_pulser_r_line2() : セクション 6.10 参照

◆　任意の２軸円弧補間

上記のように接続後、***_7443_pulser_r_arc2()***ｺﾏﾝﾄﾞによりﾊﾟﾙｻｰによる円弧補間が行えます。ﾊﾟﾗﾒｰﾀ AxisArray により動作させる軸の選択ができます。

_7443_pulser_r_arc2() : セクション 6.10 参照

4.2 モータドライバインターフェース

PPCI7443 にはサーボモータドライバのインターフェース用に INP、ALM、ERC、SVON、および RDY 信号を用意してあります。INP と ALM はサーボドライバの状態をフィードバックするのに使用します。ERC は、特別な条件のもとでサーボドライバの偏差カウンタをリセットするときに使用します。SVON は出力信号で、RDY は入力信号です。ハードウェアは事前に定義された規則に従って、INP、ALM、および ERC 信号を処理します。例えば、ALM 信号を受信したとき、PPCI7443 は自動的に出力パルスを止めて、即停止させるか減速停止させます。しかし、SVON と RDY 信号はそのケースではありません。それらは普通の入出力のように作用します。

4.2.1 ＩＮＰ

INP 信号の処理はハードウェアに事前に定義された規則で処理され、その信号は、サーボドライバのインポジション信号で動作するよう設計されています。
通常、パルス列入力のサーボモータドライバには、入力パルス指令とフィードバックカウンタとの間で、偏差検出（位置エラー）のために偏差カウンタがあります。ドライバは、最小の偏差カウンタが「0」になるまでサーボモータの動作を制御します。理論的には、サーボモータはコマンドパルスから若干の時間遅れで動作します。従って、パルス発振器がパルスを出力するのを止めても、サーボモータは停止しないで偏差カウンタが「0」になるまで動作を続けます。偏差カウンタが「0」になると、サーボドライバはモータの動作を停止させるのにパルス発振器へインポジション信号（INP）を出力します。
通常、PPCI7443 は、定められた出力パルスの完了でパルスを出力するのを止めます。しかし、***_7443_set_inp()***コマンドのパラメータ ***inp_enable*** によって、INP 信号が ON する（目標位置に到達する）までの時間、完了を遅らせることができます。***_7443_motion_done()***コマンドの状態と INT 信号もまた同様に遅れます。すなわち、位置のコントロールモードのもとに実行させるとき、***_7443_start_ta_move()、_7443_start_sr_move()***…等のファンクションの完了は、INP 信号が ON するまで遅れます。
インポジション信号は有効/無効の設定ができます。そして入力の論理も同様に、***_7443_set_inp()***コマンドのパラメータ ***inp_logic*** で設定することができます。INP 信号の状態はソフトウェア機能の***_7443_get_io_status()***コマンドによって監視することができます。

◆ 関連ファンクション:
_7443_set_inp() : セクション6.12参照
_7443_get_io_status() : セクション6.13参照
_7443_motion_done() : セクション6.11参照

4.2.2 ＡＬＭ

ALM 信号の処理はﾊｰﾄﾞｳｪｱに事前に定義された規則で処理され、その信号は、ｻｰﾎﾞﾄﾞﾗｲﾊﾞのｱﾗｰﾑ信号で動作するよう設計されています。
ALM 信号はｻｰﾎﾞﾄﾞﾗｲﾊﾞからの出力信号です。
ALM ﾋﾟﾝはｻｰﾎﾞﾄﾞﾗｲﾊﾞからｱﾗｰﾑ信号を受信します。ｱﾗｰﾑ信号は直ちに、PPCI7443 からの出力ﾊﾟﾙｽを急停止するか、減速停止させます。ALM 信号が始めから ON 状態であれば、どんなｺﾏﾝﾄﾞﾊﾟﾙｽも出力させないで、INT 信号を出力します。ALM 信号はﾊﾟﾙｽ信号であり、最小幅は５μsec です。
_7443_set_almｺﾏﾝﾄﾞのﾊﾟﾗﾒｰﾀ ***alm_logic*** で ALM 信号の入力論理を、そして、ﾊﾟﾗﾒｰﾀ ***alm_mode*** によってｱﾗｰﾑ信号入力時の停止ﾓｰﾄﾞを設定できます。
PPCI7443 がﾊﾟﾙｽを出力しているか否かに関係なく、ALM 信号は INT 信号を出力させます。ALM 信号の状態は***_7443_get_io_status()***ｺﾏﾝﾄﾞによって監視することができます。割り込み信号が有効に設定されているならば ALM 信号は IRQ を発生させることができます。(ｾｸｼｮﾝ 4.7 参照)

◆ 関連ﾌｧﾝｸｼｮﾝ:
_7443_set_alm() : セクション6.12参照
_7443_get_io_status() : セクション6.13参照

4.2.3 ＥＲＣ

ERC 信号は PPCI7443 からの出力信号です。
偏差ｶｳﾝﾀｸﾘｱ信号は下記の４つの状況で出力されます。

(1) 原点復帰の完了
(2) ｴﾝﾄﾞﾘﾐｯﾄｽｲｯﾁが ON した
(3) ｱﾗｰﾑ信号によって OUT と DIR 信号を停止した
(4) ｿﾌﾄｳｪｱ機能の非常停止ｺﾏﾝﾄﾞが発行された

ｻｰﾎﾞﾓｰﾀは PPCI7443 が発生したﾊﾟﾙｽから若干の遅れをもって運転するので、ｴﾝﾄﾞﾘﾐｯﾄｽｲｯﾁが ON するか原点復帰信号によって PPCI7443 からの出力ﾊﾟﾙｽが停止してもﾄﾞﾗｲﾊﾞの偏差ｶｳﾝﾀに残っている値が「0」になるまで動作します。ERC 信号は偏差ｶｳﾝﾀを「0」にﾘｾｯﾄすることによりｻｰﾎﾞﾓｰﾀをすぐに停止させます。ERC 信号は１ｼｮｯﾄの出力信号です。出力ﾊﾟﾙｽ幅は***_7443_set_erc()***ｺﾏﾝﾄﾞで設定できます。ERC 信号は±EL 信号か ALM 信号がすぐにｻｰﾎﾞﾓｰﾀを停止させるために ON したとき、自動的に出力されます。

◆ 関連ﾌｧﾝｸｼｮﾝ:
_7443_set_erc() : セクション6.12参照

4.2.4 ＳＶＯＮ，ＲＤＹ

PPCI7443 では、すべての軸に SVON 信号出力と RDY 信号入力があります。SVON はｻｰﾎﾞﾓｰﾀﾄﾞﾗｲﾊﾞのｻｰﾎﾞ ONｺﾏﾝﾄﾞとして実行され、そして、RDY 信号はｻｰﾎﾞﾄﾞﾗｲﾊﾞからｻｰﾎﾞﾚﾃﾞｨ信号を受信します。
SVON 信号はｿﾌﾄｳｪｱ機能の***_7443_Set_Servo()***ｺﾏﾝﾄﾞによって制御されます。
RDYﾋﾟﾝはﾃﾞｼﾞﾀﾙ入力用に用意してあります。RDY 信号の状態はｿﾌﾄｳｪｱ機能の***_7443_get_io_status()***ｺﾏﾝﾄﾞによって監視することができます。
SVON 信号は汎用出力、RDY 信号は汎用入力としても使用できます。

_7443_Set_Servo() : セクション6.12参照
_7443_get_io_status() : セクション6.13参照

4.3 ﾘﾐｯﾄｽｲｯﾁｲﾝﾀｰﾌｪｰｽと入出力状態

このセクションでは、下記の I/O 信号の操作を説明します。

- ＳＤ/ＰＣＳ： ｽﾛｰﾀﾞｳﾝ/位置変更センサ
- ±ＥＬ： エンドリミットセンサ
- ＯＲＧ： 原点センサ

どんな操作ﾓｰﾄﾞでも±EL 信号がﾓｰﾀ動作中に ON すれば、PPCI7443 は出力ﾊﾟﾙｽを自動的に停止します。ｽﾛｰﾀﾞｳﾝ信号がﾓｰﾀ動作中に ON すれば PPCI7443 は減速します。複数軸ﾓｰﾄﾞの操作であれば、関連するすべての軸に適用されます。

4.3.1 ＳＤ/ＰＣＳ

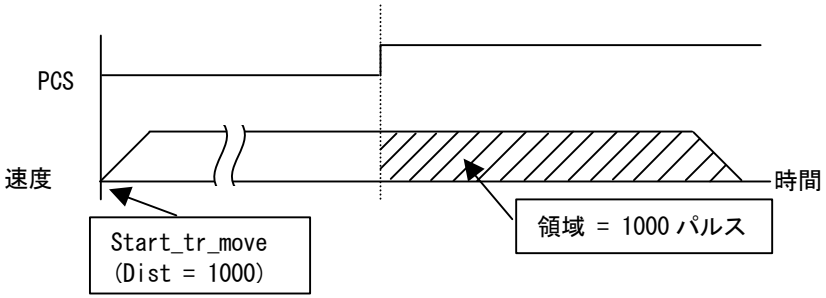
各軸の SD/PCS ﾋﾟﾝは入力信号で、SD(Slow Down)か PCS(Position Change Signal)のどちらに接続するかが選択できます。***_7443_set_sd_pin()***ｺﾏﾝﾄﾞにより SD か PCS を選択することができます。

SD/PCS ﾋﾟﾝが SD が選択(初期設定値)されたとき、PCS 信号は Low ﾚﾍﾞﾙに保たれます。そして、PCS が選択されたとき、SD 信号は Low ﾚﾍﾞﾙに保たれます。使用されない信号のために論理と有効/無効の特性に十分注意して下さい。

ｽﾛｰﾀﾞｳﾝ信号は信号が有効の時に制御出力信号(OUT,DIR)を減速させ、移動開始速度(***SteVer***)を保つときに使用します。***SteVer***は通常 ***MaxVer***よりも遅く、高速の下で機械限界に向かって動作させる場合、機械を保護するために非常に役に立ちます。SD 信号は＋方向と－方向の両方に有効です。

ｽﾛｰﾀﾞｳﾝ機能は、ｿﾌﾄｳｪｱ機能の***_7443_set_sd()***ｺﾏﾝﾄﾞにより有効/無効の設定ができます。また、入力論理、信号の保持ﾓｰﾄﾞの設定も同様にできます。信号の状態は***_7443_get_io_status()***ｺﾏﾝﾄﾞによって監視することができます。

PCS 信号は、あらかじめセットされた ***tr***、***sr*** 動作の移動開始位置を定義するのに使用されます。下記の図をご参照下さい。PCS 信号の論理は ***_7443_set_pcs_logic()***ｺﾏﾝﾄﾞで設定可能です。



◆ 関連ﾌｧﾝｸｼｮﾝ:
_7443_set_sd_pin(),_7443_set_pcs_logic() : セクション6.5参照
_7443_set_sd() : セクション6.12参照
_7443_get_io_status() : セクション6.13参照

4.3.2 ＥＬ

ｴﾝﾄﾞﾘﾐｯﾄ信号はｴﾝﾄﾞﾘﾐｯﾄｾﾝｻがON したときに、制御出力信号(OUT,DIR)を止めるのに使用されます。停止ﾓｰﾄﾞには２つのﾓｰﾄﾞがあり、一つが急停止ﾓｰﾄﾞで、もう一つが減速停止ﾓｰﾄﾞです。このﾓｰﾄﾞの選択は ***_7443_set_el()***ｺﾏﾝﾄﾞで設定できます。
+EL 信号はﾌﾟﾗｽ側のｴﾝﾄﾞﾘﾐｯﾄ信号で、-EL 信号はﾏｲﾅｽ側のｴﾝﾄﾞﾘﾐｯﾄ信号を示します。出力ﾊﾟﾙｽ信号(OUT,DIR)が正方向に出力されているとき、+EL 信号がONするとすぐに出力ﾊﾟﾙｽ信号を停止させます。出力ﾊﾟﾙｽ信号が正方向に出力されているときに-EL 信号がONしても停止しません。+EL がONして出力ﾊﾟﾙｽが完全に停止したときには、ﾓｰﾀを一方向に動かすために負方向(－)のﾊﾟﾙｽしか出力できません。
もし、割り込み信号が有効に設定されていれば、EL 信号は IRQ を発生させることができます。（ｾｸｼｮﾝ 4.7 参照）
EL 信号は、ﾃﾞｨｯﾌﾟｽｲｯﾁ S1 の設定により、Ａ接点、Ｂ接点のいづれかを選択できます。PPCI7443 のﾃﾞｨｯﾌﾟｽｲｯﾁ S1 は出荷時にはすべてのﾋﾞｯﾄがON(Ａ接点)に設定されています。
ｴﾝﾄﾞﾘﾐｯﾄ信号の状態はｿﾌﾄｳｪｱ機能の***_7443_get_ io_status()***ｺﾏﾝﾄﾞによって監視することができます。

◆ 関連ﾌｧﾝｸｼｮﾝ:
_7443_set_el() : セクション6.12参照
_7443_get_io_status() : セクション6.13参照

4.3.3 ＯＲＧ

ﾓｰﾀが原点復帰ﾓｰﾄﾞによって動作しているとき、ORG 信号は制御出力信号(OUT,DIR)を停止させるのに使用します。原点復帰ﾓｰﾄﾞには 13 種類(ｾｸｼｮﾝ 4.1.8 参照)あり、ｿﾌﾄｳｪｱ機能の***_7443_set_home_config()***ｺﾏﾝﾄﾞのﾊﾟﾗﾒｰﾀ ***home_mode*** により１つを選択できます。また、入力論理、信号の保持ﾓｰﾄﾞの設定も同様にできます。
_7443_set_home_config()ｺﾏﾝﾄﾞで原点復帰ﾓｰﾄﾞを設定した後に、
_7443_home_move()ｺﾏﾝﾄﾞで原点復帰動作を実行します。

- 関連ﾌｧﾝｸｼｮﾝ:
  _7443_set_home_config(),_7443_home_move() : セクション6.19参照

4.4 カウンタ信号

PPCI7443 は各軸に４個のカウンタがあります。そして、それらはこのセクションで説明します。

ｺﾏﾝﾄﾞｶｳﾝﾀ：	出力ﾊﾟﾙｽ(ｺﾏﾝﾄﾞﾊﾟﾙｽ)をｶｳﾝﾄする
ﾌｨｰﾄﾞﾊﾞｯｸｶｳﾝﾀ：	入力ﾊﾟﾙｽ(ﾌｨｰﾄﾞﾊﾞｯｸﾊﾟﾙｽ)をｶｳﾝﾄする
ﾎﾟｼﾞｼｮﾝｴﾗｰｶｳﾝﾀ：	ｺﾏﾝﾄﾞﾊﾟﾙｽとﾌｨｰﾄﾞﾊﾞｯｸﾊﾟﾙｽの差をｶｳﾝﾄ
汎用ｶｳﾝﾀ：	ﾊﾟﾙｽ出力、ﾌｨｰﾄﾞﾊﾞｯｸﾊﾟﾙｽ、手動ﾊﾟﾙｻｰまたは CLK/2 としてｶｳﾝﾀ入力を構成する

また、ﾀｰｹﾞｯﾄﾎﾟｼﾞｼｮﾝﾚｺｰﾀﾞ(ｿﾌﾄｳｪｱで維持された位置のﾚｺｰﾀﾞ)について説明します。

4.4.1 ｺﾏﾝﾄﾞｶｳﾝﾀ

ｺﾏﾝﾄﾞｶｳﾝﾀは 28 ﾋﾞｯﾄのﾊﾞｲﾅﾘｰ UP/DOWN ｶｳﾝﾀで、入力源は、PPCI7443 からの出力ﾊﾟﾙｽです。そして、それは現在のｺﾏﾝﾄﾞﾎﾟｼﾞｼｮﾝの正確な情報として提供します。　注意:ｺﾏﾝﾄﾞﾎﾟｼﾞｼｮﾝはﾀｰｹﾞｯﾄﾎﾟｼﾞｼｮﾝと異なっています。ｺﾏﾝﾄﾞﾎﾟｼﾞｼｮﾝはﾀｰｹﾞｯﾄﾎﾟｼﾞｼｮﾝが新しいｺﾏﾝﾄﾞを実行したときだけ変化するのに対して、ﾊﾟﾙｽ出力により増減します。ﾀｰｹﾞｯﾄﾎﾟｼﾞｼｮﾝはｿﾌﾄｳｪｱで表示し、原点復帰完了後に手動でﾘｾｯﾄするのが必要です。
ｺﾏﾝﾄﾞﾎﾟｼﾞｼｮﾝのｶｳﾝﾀは原点復帰動作完了で自動的に「0」にﾘｾｯﾄします。更に、***_7443_set_command()***ｺﾏﾝﾄﾞにより、いつでも新しいｺﾏﾝﾄﾞﾎﾟｼﾞｼｮﾝの値を設定することができます。***_7443_get_command()***ｺﾏﾝﾄﾞにより現在のｺﾏﾝﾄﾞﾎﾟｼﾞｼｮﾝを読み込みます。

◆ 関連ﾌｧﾝｸｼｮﾝ:

_7443_set_command(),_7443_get_command() : セクション6.15参照

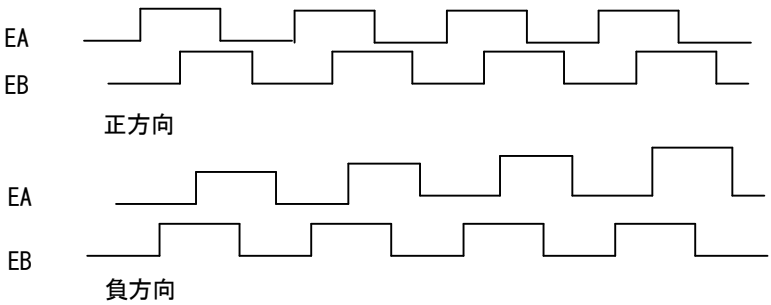
4.4.2 ﾌｨｰﾄﾞﾊﾞｯｸｶｳﾝﾀ

PPCI7443 は各軸のために現在のﾌｨｰﾄﾞﾊﾞｯｸ位置を管理するための 28 ﾋﾞｯﾄのﾊﾞｲﾅﾘｰ UP/DOWN ｶｳﾝﾀがあります。ｶｳﾝﾀは EA と EB ﾋﾟﾝから入力された信号をｶｳﾝﾄします。
それは２種類のﾊﾟﾙｽ入力を受け取ることができます。一つがﾌﾟﾗｽとﾏｲﾅｽのﾊﾟﾙｽ入力ﾓｰﾄﾞ(CW/CCW)で、もう一つが 90 度位相差信号ﾓｰﾄﾞ(AB 相)です。90 度位相差ﾓｰﾄﾞには１倍、２倍、４倍のﾓｰﾄﾞが選択できます。４倍位相差ﾓｰﾄﾞがｲﾝｸﾘﾒﾝﾀﾙｴﾝｺｰﾀﾞ入力に最も一般的に使用されています。例えば、１回転あたり 2000 ﾊﾟﾙｽの回転式ｴﾝｺｰﾀﾞの場合４倍ﾓｰﾄﾞだとするとｶｳﾝﾀに入力される値は 8000 ﾊﾟﾙｽか-8000 ﾊﾟﾙｽになります。ﾌﾟﾗｽかﾏｲﾅｽはﾓｰﾀの回転方向によります。***_7443_set_pls_iptmode()***ｺﾏﾝﾄﾞはこれらの入力ﾓｰﾄﾞを選択します。

ｴﾝｺｰﾀﾞを使用しないｱﾌﾟﾘｹｰｼｮﾝの場合、ﾌｨｰﾄﾞﾊﾞｯｸｶｳﾝﾀ入力をｺﾏﾝﾄﾞｶｳﾝﾀと同じように出力ﾊﾟﾙｽに設定することができます。このとき、ﾌｨｰﾄﾞﾊﾞｯｸｶｳﾝﾀとｺﾏﾝﾄﾞｶｳﾝﾀは同じ値になります。出力ﾊﾟﾙｽをｶｳﾝﾄするには ***_7443_set_feedback_src()***ｺﾏﾝﾄﾞのﾊﾟﾗﾒｰﾀ ***Src*** を「1」に設定します。

◆ ﾌﾟﾗｽとﾏｲﾅｽﾊﾟﾙｽの入力ﾓｰﾄﾞ　　(CW/CCW ﾓｰﾄﾞ)
このﾓｰﾄﾞにおけるﾊﾟﾙｽのﾊﾟﾀｰﾝはﾊﾟﾙｽ指令のｾｸｼｮﾝ 4.1.1 の２ﾊﾟﾙｽ出力ﾓｰﾄﾞ(CW/CCW ﾓｰﾄﾞ)と同じであり、入力ﾋﾟﾝが EA と EB であると思って下さい。
このﾓｰﾄﾞでは、EA ﾋﾟﾝからの入力ﾊﾟﾙｽはｶｳﾝﾄｱｯﾌﾟ(＋)でｶｳﾝﾄし、EB ﾋﾟﾝからの入力ﾊﾟﾙｽはｶｳﾝﾄﾀﾞｳﾝ(－)としてｶｳﾝﾄします。

◆ 90° の位相差信号入力ﾓｰﾄﾞ(AB 相ﾓｰﾄﾞ)
このﾓｰﾄﾞでは、EA 信号は EB 信号に比べ 90 度進むか遅れます。２つの信号間の位相差の「進み」か「遅れ」はﾓｰﾀの回転方向によって決められます。EA 信号が EB 信号より位相が進んでいるとき、ｱｯﾌﾟﾀﾞｳﾝｶｳﾝﾀはｶｳﾝﾄｱｯﾌﾟします。
下記の図は EA、EB 信号の波形を示します。



ｴﾝｺｰﾀﾞからの EZ 相入力(EZ)信号は EZ 相が「0」になったのを入力するのに使用されます。この信号は回転ﾓｰﾀの大部分で共通です。EZ 信号は機械のｱﾌﾞｿﾘｭｰﾄ位置を定義するのに使用することができます。EZ 信号の入力論理は ***_7443_set_home_config()***ｺﾏﾝﾄﾞによって設定できます。EZ 信号の状態を ***_7443_get_io_status()***ｺﾏﾝﾄﾞで監視することができます。
ﾌｨｰﾄﾞﾊﾞｯｸｶｳﾝﾀは原点復帰動作完了後、自動的に「0」にｸﾘｱされます。更に、***_7443_set_position()***ｺﾏﾝﾄﾞにより、いつでも新しいｺﾏﾝﾄﾞﾎﾟｼﾞｼｮﾝの値を設定することができます。***_7443_get_position()***ｺﾏﾝﾄﾞにより現在のﾌｨｰﾄﾞﾊﾞｯｸﾎﾟｼﾞｼｮﾝを読み込みます。

◆ 関連ﾌｧﾝｸｼｮﾝ：
_7443_set_pls_iptmode(),_7443_set_feedback_src()：セクション6.4参照
_7443_set_position(),_7443_get_position()：セクション6.15参照
_7443_set_home_config()：セクション6.9参照

4.4.3 ﾎﾟｼﾞｼｮﾝｴﾗｰｶｳﾝﾀ

ﾎﾟｼﾞｼｮﾝｴﾗｰｶｳﾝﾀは、ｺﾏﾝﾄﾞﾎﾟｼﾞｼｮﾝとﾌｨｰﾄﾞﾊﾞｯｸﾎﾟｼﾞｼｮﾝ間での誤差について計算するのに使用されます。動作理論はPPCI7443が1ﾊﾟﾙｽを出力したら1つのｶｳﾝﾄを加え、PPCI7443が1ﾊﾟﾙｽ受信したとき(EA,EBから)1つのｶｳﾝﾄを引き算するということです。ｴﾝｺｰﾀﾞが適用されているとき、ｽﾃｯﾋﾟﾝｸﾞﾓｰﾀの脱調検出に使用可能です。
ﾎﾟｼﾞｼｮﾝｴﾗｰｶｳﾝﾀは出力ﾊﾟﾙｽとﾌｨｰﾄﾞﾊﾞｯｸﾊﾟﾙｽの差を自動的に計算するので、その動作比率が異なればエラーとなります。
_7443_get_error_counter()ｺﾏﾝﾄﾞでﾎﾟｼﾞｼｮﾝｴﾗｰｶｳﾝﾀの値を読み出します。
_7443_reset_error_counter()ｺﾏﾝﾄﾞでﾎﾟｼﾞｼｮﾝｴﾗｰｶｳﾝﾀの値をﾘｾｯﾄします。
ﾎﾟｼﾞｼｮﾝｴﾗｰｶｳﾝﾀの値は原点動作完了後、自動的に「0」にｸﾘｱされます。

◆ 関連ﾌｧﾝｸｼｮﾝ：
_7443_get_error_counter(),_7443_reset_error_counter()：セクション6.15参照

4.4.4 汎用ｶｳﾝﾀ

汎用ｶｳﾝﾀのｶｳﾝﾀ入力は最も万能で、それは下記の通りです。

1. ﾊﾟﾙｽ出力： ｺﾏﾝﾄﾞﾎﾟｼﾞｼｮﾝｶｳﾝﾀと同じ
2. ﾊﾟﾙｽ入力： ﾌｨｰﾄﾞﾊﾞｯｸｶｳﾝﾀと同じ
3. ﾊﾟﾙｻｰ入力： 初期状態
4. ｸﾛｯｸ： 正確なﾀｲﾏｰ(9.8MHz)

汎用ｶｳﾝﾀのｶｳﾝﾀ入力の初期設定は手動ﾊﾟﾙｻｰです。(手動ﾊﾟﾙｻｰに関する詳細説明についてはｾｸｼｮﾝ4.1.9をご参照下さい)他のｶｳﾝﾀ入力を設定するには***_7443_set_general_counter()***ｺﾏﾝﾄﾞで設定できます。そして、ｶｳﾝﾀ値の読み出しは***_7443_get_general_counter()***ｺﾏﾝﾄﾞでできます。

◆ 関連ﾌｧﾝｸｼｮﾝ：
_7443_set_general_counter(),_7443_get_general_counter()：セクション6.15参照

カウンタ	記述	カウンタ入力	ファンクションライブラリ	機能記述
コマンド ポジション カウンタ	出力したパルス数のカウント	出力パルス	***_7443_set_command***	コマンドポジションに新しい値を設定する
			_7443_get_command	現在のコマンドポジションを読み込む
フィードバック カウンタ	入力したパルス数のカウント	出力パルス または EA/EB	***_7443_set_pls_iptmode***	EA/EB の入力モードを選択する
			_7443_set_feedback_src	カウンタ入力を設定する
			_7443_set_position	フィードバックポジションに新しい値を設定する
			_7443_get_position	現在のフィードバックポジションを読み込む
ポジション エラーカウンタ	コマンドパルスとフィードバックパルスの差	出力パルスと EA/EB	***_7443_get_error_counter***	ポジションエラーカウンタの値を読み込む
			_7443_reset_error_counter	ポジションエラーカウンタの値をリセットする
汎用カウンタ	汎用目的のカウンタ	出力パルス EA/EB 手動パルサー 基本 CLK÷2	***_7443_set_general_counter***	新しいカウンタ値を設定する。
			_7443_get_general_counter	現在のカウンタ値を読み込む

4.4.5 ﾀｰｹﾞｯﾄﾎﾟｼﾞｼｮﾝﾚｺｰﾀﾞ

例えば、PPCI7443 がｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞで連続した動作をしている時、次のｱﾌﾞｿﾘｭｰﾄ動作の目標位置は以前の目標位置を記憶しています。

ﾀｰｹﾞｯﾄﾎﾟｼﾞｼｮﾝﾚｺｰﾀﾞがｿﾌﾄｳｪｱによって処理されるのを理解するのは非常に重要です。新しい動作ｺﾏﾝﾄﾞが実行される度に置換えは自動的にﾀｰｹﾞｯﾄﾎﾟｼﾞｼｮﾝﾚｺｰﾀﾞに加算されます。そして、ﾀｰｹﾞｯﾄﾎﾟｼﾞｼｮﾝﾚｺｰﾀﾞの正当性を確かにするために、下記の２つの状況で***_7443_reset_target_pos()***ｺﾏﾝﾄﾞを呼び出すことによって手動にて維持する必要があります。

1. 原点復帰動作完了後
2. 新しいﾌｨｰﾄﾞﾊﾞｯｸﾎﾟｼﾞｼｮﾝを設定した後

◆ 関連ﾌｧﾝｸｼｮﾝ:
_7443_reset_target_pos() : セクション6.15参照

4.5　複数のPPCI7443の操作

ｿﾌﾄｳｪｱ機能ﾗｲﾌﾞﾗﾘは最大 12 枚の PPCI7443 をｻﾎﾟｰﾄでき、最大 48 軸のﾓｰﾀの制御が可能です。PPCI7443 はﾌﾟﾗｸﾞ&ﾌﾟﾚｰで使用できるので、基板のﾍﾞｰｽｱﾄﾞﾚｽ、IRQ ﾚﾍﾞﾙを設定する必要はありません。ﾍﾞｰｽｱﾄﾞﾚｽ等の設定はｼｽﾃﾑを立ち上げる時にｼｽﾃﾑの BIOS によって自動的に割り当てられます。挿入された PPCI7443 が正しくｲﾝｽﾄｰﾙされ、Base ｱﾄﾞﾚｽと IRQ ﾚﾍﾞﾙが BIOS によって割り当てられるていることを「PPCI7443 Utility」を利用してﾁｪｯｸできます。

複数の基板を使用しているとき、識別された基板のｶｰﾄﾞ番号を確認する必要があります。PPCI7443 のｶｰﾄﾞ番号は PCI ｽﾛｯﾄの位置に依存します。それらの番号は PCI ｽﾛｯﾄの右から左、もしくは左から右に割り付けられます。これらのｶｰﾄﾞ番号は基板の対応する軸番号に作用します。そして、軸の番号はﾗｲﾌﾞﾗﾘの中の殆どのｺﾏﾝﾄﾞで最初のﾊﾟﾗﾒｰﾀとして使用されます。したがって、ｱﾌﾟﾘｹｰｼｮﾝﾌﾟﾛｸﾞﾗﾑを書く前に軸の番号を確認するのは重要です。例えば、３枚の PPCI7443 基板が PCI ｽﾛｯﾄに挿入されていれば、各基板に対応する軸の番号は下記の表の通りです。

軸 No. / 基板 No.	１軸目	２軸目	３軸目	４軸目
０ (1 枚目)	０	１	２	３
１ (2 枚目)	４	５	６	７
２ (3 枚目)	８	９	１０	１１

もし一定速度ﾓｰﾄﾞで２枚目の基板の３軸目を 0.5 秒後に 0pps から 10000pps まで速度を加速させたければ、軸番号は６で、ﾌﾟﾛｸﾞﾗﾑｺｰﾄﾞは次の通りである。

```
_7443_start_tv_move(6,0,10000,0.5);
```

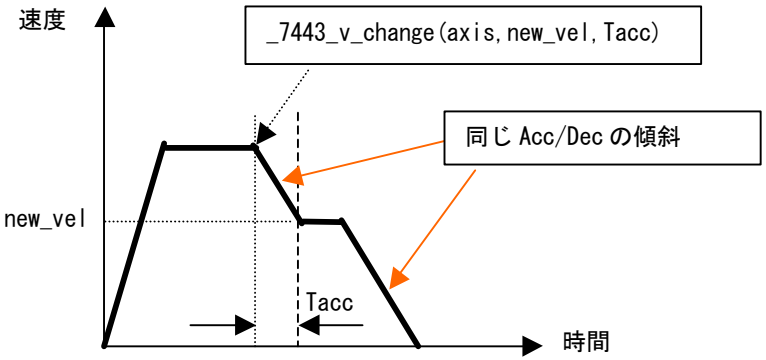
正しい基板番号を確認するために、ﾃｽﾄとｴﾗｰをｱﾌﾟﾘｹｰｼｮﾝの開発前に行う必要があります。「PPCI7443 Utility」は基板番号の検索時間を最小にするのに利用できます。

複数の PPCI7443 の複数軸を同時に動作させるｱﾌﾟﾘｹｰｼｮﾝが必要であるならば、ｾｸｼｮﾝ 3.12 の接続図の様に CN4 ｺﾈｸﾀを配線して下さい。このようなｱﾌﾟﾘｹｰｼｮﾝのﾌﾟﾛｸﾞﾗﾑを書くとき、ｾｸｼｮﾝ 6.8 で示したいくつかのｺﾏﾝﾄﾞが役に立ちます。

4.6　オーバーライド機能（速度、位置）

PPCI7443 は軸の動作中に速度もしくは位置を変更することができます。

4.6.1 速度のオーバーライド



速度のオーバーライドは１軸の動作の時だけ適用できます。アブソリュートモード動作、インクリメンタルモード動作両方とも適用できます。上のグラフは基本的な操作理論を示します。
下記のファンクションは速度のオーバーライド機能に関連します。

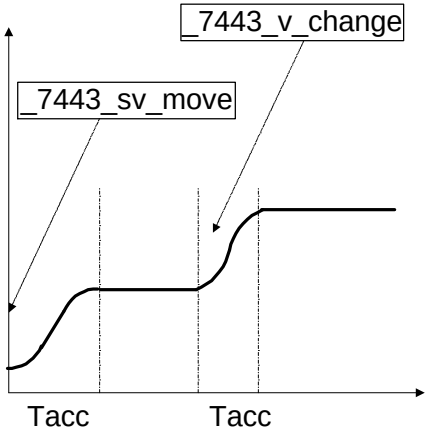
_7443_v_change()：	速度のオーバーライド（MaxVel を変える）
_7443_cmp_v_change()：	汎用コンパレータが成立すると変える
_7443_sd_stop()：	減速停止する
_7443_emg_stop()：	急停止する
_7443_fix_speed_range()：	速度範囲を定義する
_7443_unfix_speed_range()：	速度範囲の抑制の解除

最初の４つのファンクションは１軸の動作中に、速度を変化させることができます。しかし、***_7443_sd_stop()***コマンドと***_7443_emg_stop()***コマンドは軸の速度を「0」に変えます（停止）。***_7443_fix_speed_range()***コマンドは***_7443_v_change()***コマンドより前に必要で、**_7443_unfix_speed_range()**コマンドは***_7443_fix_speed_range()***コマンドによる速度制限を解除します。***_7443_cmp_change()***コマンドは汎用コンパレータが成立したときだけで、***_7443_cmp_change()***コマンドが作用する事を除いて***_7443_v_change()***コマンドと殆ど同じファンクションです。汎用コンパレータに関するその他の詳細についてはセクション 4.4.4 をご参照下さい。

最後の４つのﾌｧﾝｸｼｮﾝは比較的分かり易く、使用できます。したがって、説明は***_7443_v_change()***にします。

***_7443_v_change()*の理論を扱う:**

_7443_v_change()ｺﾏﾝﾄﾞは、その場その場で ***MaxVel*** を変えるのに使用されます。通常の動作では、軸は ***StrVel*** 速度で始まって ***MaxVel*** に加速し、次に減速領域に達するまで ***MaxVel*** を保ちます。もし ***MaxVel*** を変えるならば、定義された期間に強制的に新しい速度まで加速するか、または減速させます。台形動作、Ｓ字曲線動作の両方で適用可能です。変化速度は台形動作そして、Ｓ字曲線動作の一定の加速性です。



Ｓ字曲線がある速度変化

***_7443_v_change()*の抑制:**

１軸のﾌﾟﾘｾｯﾄﾓｰﾄﾞでは、新しい速度に達することができる十分な残りﾊﾟﾙｽがなければなりません。もしそうでなければ、エラーを返して、***_7443_v_change()***ｺﾏﾝﾄﾞは速度を変化させずに保つでしょう。

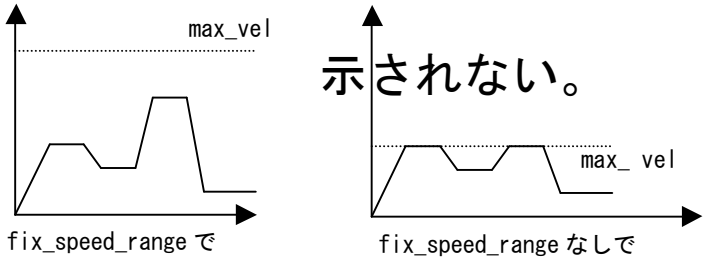
例えばｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでの台形動作を実行する:

```
    _7443_start_tr_move(0,10000,0,1000,0.1,0.1);
```

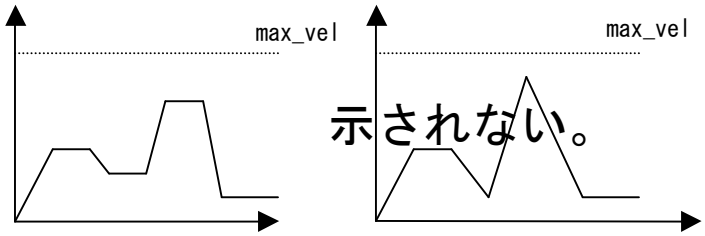
これは軸 0 の移動距離が 10000 ﾊﾟﾙｽで、最高速度が 1000ppsPPS である。***_7443_v_change***(0,NewVel,Tacc)では 5000 ﾊﾟﾙｽが適用可能です。

NewVel (pps)	Tacc (秒)	必要な残りパルス			OK/ｴﾗｰ
		加速	減速	合計	
5000	0.1	300	313	613	OK
5000	1	3000	3125	6125	ｴﾗｰ
10000	0.1	550	556	1106	OK
50000	0.1	2550	2551	5101	ｴﾗｰ

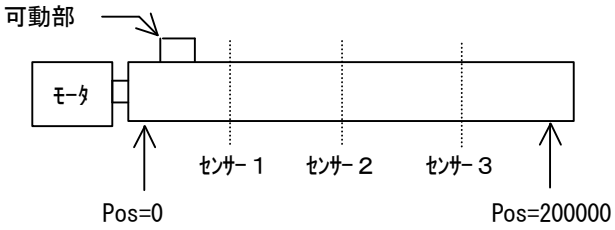
1. ***_7443_v_change()***ｺﾏﾝﾄﾞが正しく動作するように、***_7443_fix_speed_range()***ｺﾏﾝﾄﾞで最高速度を設定しなければなりません。もしそうでなければ、***_7443_v_move()***ｺﾏﾝﾄﾞか***_7443_start_ta_move()***ｺﾏﾝﾄﾞによって設定された***MaxVel***は自動的に***_7443_v_change()***ｺﾏﾝﾄﾞでえることができない最高速度になります。



2. 加速か減速期間の***_7443_v_change()***ｺﾏﾝﾄﾞは示されない。殆どの場合動作しているが、加速/減速時間は保証されません。



例:
ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞで 200000ﾊﾟﾙｽ移動する間に３個の速度変化用ｾﾝｻｰがあります。最初の最高速度は 10000pps で、もしｾﾝｻｰ１が ON したら最高速度を 25000pps に変え、ｾﾝｻｰ２が ON したら最高速度を 50000pps に、ｾﾝｻｰ３が ON したら最高速度を 100000pps に変えます。このときのｱﾌﾟﾘｹｰｼｮﾝﾌﾟﾛｸﾞﾗﾑと速度ﾌﾟﾛﾌｨｰﾙの結果は次のように示されます。

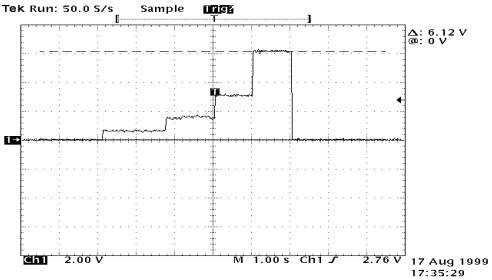



```
#include "ppci7443.h"

  _7443_fix_speed_range(axis,100000.0);
  _7443_start_ta_move(axis,200000.0,1000,10000,0.02,0.01);
  while(!_7443_motion_done(axis))
  {
                              //他の入出力ﾎﾞｰﾄﾞからｾﾝｻｰの情報を得る


       If((Sensor1==High)&&(Sensor2==Low)&&(Sensor3==Low))
                              _7443_v_change(axis,25000,0.02);
       else if((Sensor1==Low)&&(Sensor2==High)&&(Sensor3==Low))
                              _7443_v_change(axis,50000,0.02);
       else if((Sensor1==Low)&&(Sensor2==Low)&&(Sensor3==High))
                              _7443_v_change(axis,100000,0.02);
  }
```

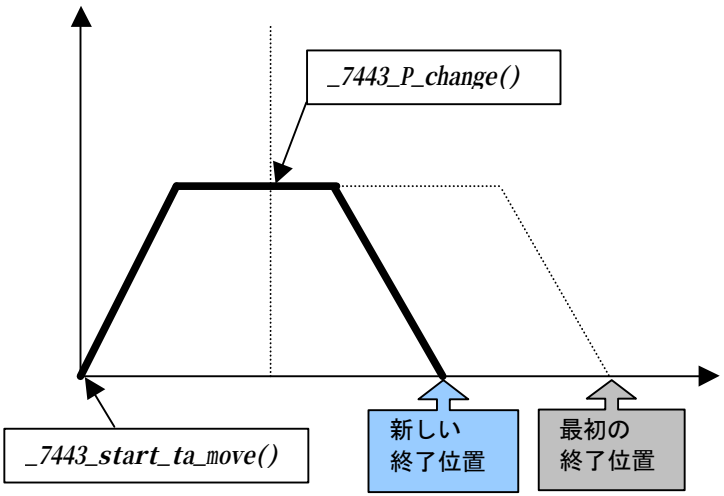
３個のｾﾝｻｰの情報が他の入出力ﾎﾞｰﾄﾞから取得される。そして、実験の結果として起こる速度ﾌﾟﾛﾌｨｰﾙは下記に示されます。



◆ 関連ﾌｧﾝｸｼｮﾝ：
_7443_v_change(),_7443_sd_stop(),_7443_emg_stop(),
_7443_fix_speed_range(),_7443_unfix_speed_range(),
_7443_get_currebt_speed()：セクション6.5参照

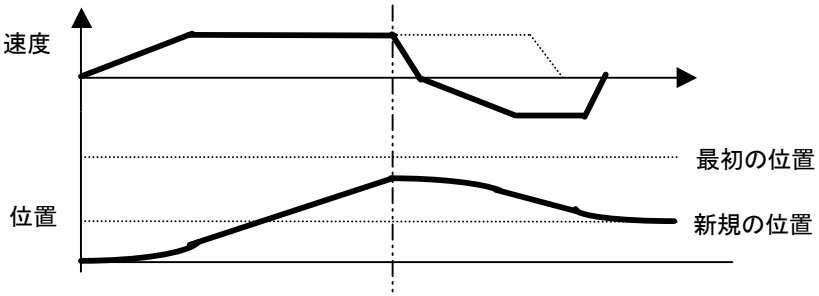
4.6.2 位置のｵｰﾊﾞｰﾗｲﾄﾞ

ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞで１軸のﾌﾟﾘｾｯﾄﾓｰﾄﾞを実行するとき、ﾓｰﾀが動作中に ***_7443_p_change()***ｺﾏﾝﾄﾞで目標位置を変更することが可能です。



***_7443_p_change()*の理論を扱う:**

_7443_p_change()ｺﾏﾝﾄﾞは***_7443_start_ta_move()***ｺﾏﾝﾄﾞと ***_7443_start_sa_move()***ｺﾏﾝﾄﾞのときだけで適用可能です。それは元々これら２つのﾌｧﾝｸｼｮﾝによって定義された目標位置を変更することになっています。位置を変更後、軸は新しい目標位置に動作し、元の目標位置を完全に無視します。もし新しい位置が通過した経路にあるならば、軸を減速停止させ、逆転させます。（下記のｸﾞﾗﾌ参照） 加速/減速、***StrVel***、および ***MaxVel*** は元の設定と同じままです。



***_7443_p_change()*の抑制:**

1. ***_7443_p_change()***ｺﾏﾝﾄﾞはｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでの１軸のﾌﾟﾘｾｯﾄﾓｰﾄﾞを実行していいるときのみ適用可能で、すなわち、***_7443_start_ta_move()***ｺﾏﾝﾄﾞおよび***_7443_start_sa_move()***ｺﾏﾝﾄﾞだけ適用可能です。
2. 減速期間での位置変更はできません。
3. ***_7443_p_change()***ｺﾏﾝﾄﾞが実行された場合、現在の位置と新しい目標位置の間には、十分な距離が必要です。なぜなら PPCI7443 は動作終了の減速に十分な領域を必要とします。

例えばｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞの台形動作を実行する:

```
_7443_start_ta_move(0,10000,0,1000,0.5,1);
```

これは軸０を 10000 の位置に動作させることで、最高速度は 1000pps です。減速するﾊﾟﾙｽの必要な数は 0.5*1000*1=500 です。

_7443_p_change(0,NewPos)では位置の「現在位置」が適用可能です。

NewPos	現在位置	OK/ｴﾗｰ	注意
5000	4000	OK	
5000	4501	ｴﾗｰ	
5000	5000	ｴﾗｰ	
5000	5499	ｴﾗｰ	
5000	6000	OK	戻る
5000	9499	OK	戻る
5000	9500	ｴﾗｰ	
5000	9999	ｴﾗｰ	

- 関連ﾌｧﾝｸｼｮﾝ：
  _7443_p_change()：セクション6.6参照

4.7　位置の比較とラッチ

PPCI7443 は軸０と軸１で位置比較機能を、そして軸２と軸３で位置ﾗｯﾁ機能を提供します。比較機能はｶｳﾝﾀが設定値に到達したときにﾄﾘｶﾞｰﾊﾟﾙｽを出力します。CMP1(軸０)と CMP2(軸１)が比較出力に使用されます。ﾗｯﾁ機能はﾗｯﾁ信号が動作することにより４つのｶｳﾝﾀ(ｾｸｼｮﾝ 4.4 参照)の値を獲得します。LTC3(軸２)と LTC4(軸３)がﾗｯﾁﾊﾟﾙｽの入力に使用されます。

4.7.1 PPCI7443のｺﾝﾊﾟﾚｰﾀ

PPCI7443 の各軸には５個のｺﾝﾊﾟﾚｰﾀがあります。各ｺﾝﾊﾟﾚｰﾀの解説表を下記に表示します。

	比較源	記述	関連ﾌｧﾝｸｼｮﾝ
ｺﾝﾊﾟﾚｰﾀ１	ｺﾏﾝﾄﾞﾎﾟｼﾞｼｮﾝｶｳﾝﾀ	ｿﾌﾄﾘﾐｯﾄ(+) (ｾｸｼｮﾝ 4.9 参照)	_7443_set_softlimit _7443_enable_softlimit _7443_diable_softlimit
ｺﾝﾊﾟﾚｰﾀ２	ｺﾏﾝﾄﾞﾎﾟｼﾞｼｮﾝｶｳﾝﾀ	ｿﾌﾄﾘﾐｯﾄ(-) (ｾｸｼｮﾝ 4.9 参照)	
ｺﾝﾊﾟﾚｰﾀ３	ﾎﾟｼﾞｼｮﾝｴﾗｰｶｳﾝﾀ	脱調検出	_7443_error_counter_check
ｺﾝﾊﾟﾚｰﾀ４	すべてのｶｳﾝﾀ	汎用目的	_7443_set_general_comparator
ｺﾝﾊﾟﾚｰﾀ５ (軸 0,1 のみ)	ﾌｨｰﾄﾞﾊﾞｯｸｶｳﾝﾀ	位置比較機能 (ﾄﾘｶﾞｰ)	_7443_set_trigger_comparator _7443_build_compare_function _7443_build_compare_table _7443_set_auto_compare

注意：すべてのｺﾝﾊﾟﾚｰﾀが CMP を通してﾄﾘｶﾞｰﾊﾟﾙｽ出力の機能を持っているというわけではありません。それはｺﾝﾊﾟﾚｰﾀ５だけです。

ｺﾝﾊﾟﾚｰﾀ１と２はｿﾌﾄﾘﾐｯﾄ(ｾｸｼｮﾝ 4.9 参照)のためです。ｺﾝﾊﾟﾚｰﾀ３は、ﾎﾟｼﾞｼｮﾝｴﾗｰｶｳﾝﾀと比較するのに使用されます。ｽﾃｯﾋﾟﾝｸﾞﾓｰﾀの脱調を検出するのに非常に役に立ちます。脱調検出の有効/無効と許容値の設定が ***_7443_set_error_counter_check()***ｺﾏﾝﾄﾞで設定できます。
もし脱調検出が有効に設定されていて検出されたならば、PPCI7443 は割り込み信号を発生させます。
ｺﾝﾊﾟﾚｰﾀ４は汎用目的のｺﾝﾊﾟﾚｰﾀで、比較状態が成立したならば、そのｺﾝﾊﾟﾚｰﾀは割り込みを発生させます。比較源はどのｶｳﾝﾀでも可能です。比較された値、比較源、比較方法および処理はｿﾌﾄｳｪｱ機能の ***_7443_set_general_comparator()***ｺﾏﾝﾄﾞによって設定されます。

4.7.2 位置の比較

位置比較機能は、比較源がﾌｨｰﾄﾞﾊﾞｯｸｶｳﾝﾀである５番目のｺﾝﾊﾟﾚｰﾀによって実行されます。最初の２軸(軸０と軸１)だけが位置比較機能を実行できます。位置比較機能は比較条件が成立したとき CMP ﾋﾟﾝよりﾄﾘｶﾞｰﾊﾟﾙｽを出力します。
比較条件は２つの部品からなり、最初の一つが比較される値で、もう一つが比較ﾓｰﾄﾞです。比較ﾓｰﾄﾞは「>」, 「=」, 「<」が可能です。位置比較機能を使う最も簡単な方法はｿﾌﾄｳｪｱ機能を呼び出すことです。

```
_7443_set_trigger_comparator(AxisNo,Method,Data)
```

２番目のﾊﾟﾗﾒｰﾀ ***Method*** は比較方法を示し、３番目の ***Data*** は比較される値です。連続した比較では、他のﾌｧﾝｸｼｮﾝで設定された比較ﾃﾞｰﾀを使用するのでこのﾃﾞｰﾀは自動的に無視されます。

◆ 連続した比較

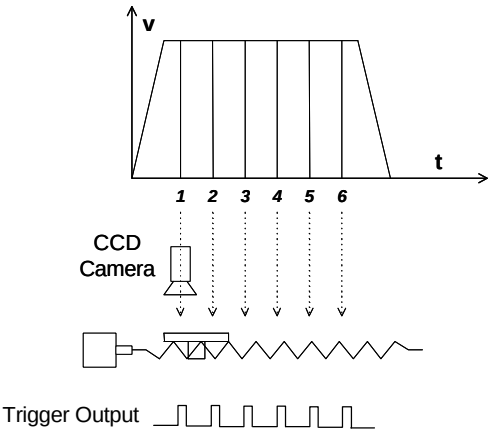
連続的に多くのﾃﾞｰﾀを比較するﾕｰｻﾞｰにとって、比較ﾃｰﾌﾞﾙを構成するﾌｧﾝｸｼｮﾝは下記に示されるように提供されます。

1. _7443_build_comp_function(AxisNo,Start,End,Interval, device)
2. _7443_build_comp_table(AxisNo,tableArray,Size,device)
3. _7443_set_auto_compare(AxisNo,SelectSource)

注意：この機能が使用されているとき、すべての割り込み機能を OFF にして下さい

最初のﾌｧﾝｸｼｮﾝは開始位置、終了位置そして比較間隔によって比較ﾘｽﾄを作ることです。２番目のﾌｧﾝｸｼｮﾝは任意の比較ﾃｰﾌﾞﾙ(ﾃﾞｰﾀ配列)を作ることです。３番目のﾌｧﾝｸｼｮﾝは比較源を選択する機能です。そして比較に FIFO ﾓｰﾄﾞを使用するためにこのﾊﾟﾗﾒｰﾀに「1」を設定して下さい。それにより比較が自動的に開始されます。***_7443_check_compare_data()***ｺﾏﾝﾄﾞは現在の比較ﾃﾞｰﾀをﾁｪｯｸします。

ここに、連続した位置の比較機能を使用する例があります。



このｱﾌﾟﾘｹｰｼｮﾝでは、ﾃｰﾌﾞﾙは動作ｺﾏﾝﾄﾞによって制御され、そして CCD ｶﾒﾗは PPCI7443 の位置比較出力によって制御されます。
動く対称物のｲﾒｰｼﾞが容易にこの方法で見る事ができます。
動作仕様：１ストロークに 34000 ﾎﾟｲﾝﾄのトリガー出力
　　トリガー出力速度は１秒間に 6000 ﾎﾟｲﾝﾄ
ﾌﾟﾛｸﾞﾗﾑは以下のように設定します。

- ﾃｰﾌﾞﾙは 0 から 36000 移動開始
- 比較ﾎﾟｲﾝﾄは 1001 から 35000 まで 1 ﾊﾟﾙｽ間隔で 34000 ﾎﾟｲﾝﾄ
- ﾃｰﾌﾞﾙの移動速度は 6000pps
- 比較条件のﾌﾟﾛｸﾞﾗﾑｺｰﾄﾞは「=」

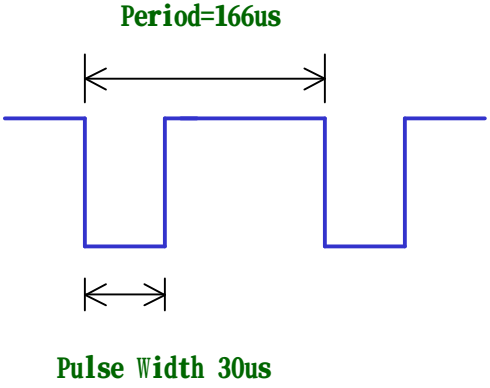
```
_7443_set_trigger_comparator(0,1,1,1001);
_7443_build_compare_function(0,1001,35000,1,1);
_7443_set_auto_compare(0,1);
_7443_start_tr_move(Axis0,36000,0,6000,0.01,0.01);
```

現在の比較ﾃﾞｰﾀのﾁｪｯｸまたは監視は

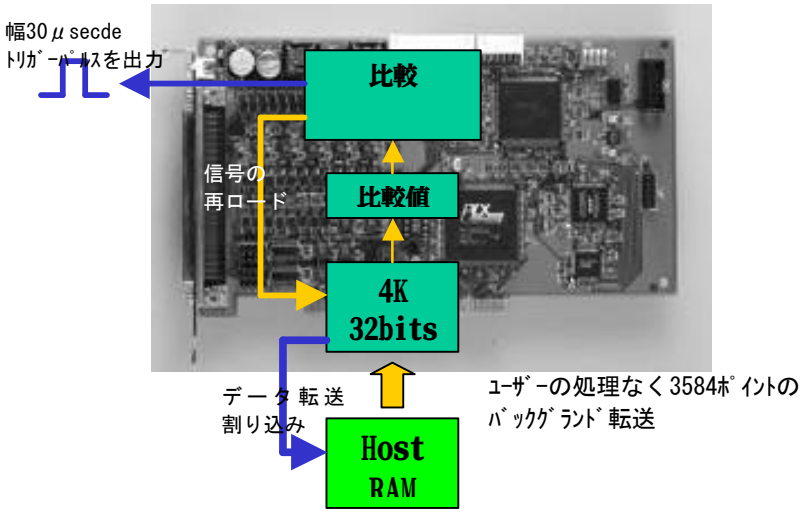
```
_7443_check_compare_data(0,5,*CurrentData);
```

自動で比較実行されてれば、この機能を使用することができます。

結果：



比較の仕組みを以下に示します。



上記の図で比較値ﾌﾞﾛｯｸは比較される位置を意味し、それをﾁｪｯｸするために_7443_check_compare_data()ｺﾏﾝﾄﾞを使用することができます。最後の比較ﾎﾟｲﾝﾄが比較された後、比較値ﾌﾞﾛｯｸには最終比較ﾎﾟｲﾝﾄの後の値を表示するのに注意して下さい。比較ﾃｰﾌﾞﾙの中に１個ダミーの比較値（移動するｽﾄﾛｰｸで動作できない値）を作成して下さい。もし、_7443_compare_function()ｺﾏﾝﾄﾞを使用した場合は、自動的にダミー比較

値を読み込みます。
この値は（最終ﾎﾟｲﾝﾄ＋ｲﾝﾀｰﾊﾞﾙ×全体のｶｳﾝﾄ）×移動の比率です。

◆ 関連ﾌｧﾝｸｼｮﾝ：
_7443_set_trigger_comparator(),_7443_build_comp_function(), _7443_build_comp_table(),_7443_set_auto_compare(), _7443_check_compare_data(),_7443_set_trigger_type()：セクション6.16参照

4.7.3 位置のラッチ

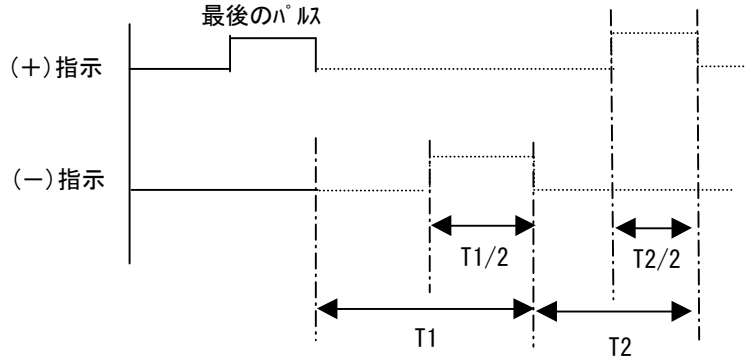
位置ﾗｯﾁは位置比較と反対の機能です。位置比較機能は比較条件が成立したとき、CMPﾋﾟﾝよりﾄﾘｶﾞｰﾊﾟﾙｽを出力します。しかし、位置のﾗｯﾁ機能はLTCﾋﾟﾝよりﾊﾟﾙｽ信号を受信し、その瞬間にｶｳﾝﾀ(ｾｸｼｮﾝ4.4参照)のﾃﾞｰﾀを獲得します。ﾗｯﾁ方法はﾊｰﾄﾞｳｪｱで作られているため、ﾗｯﾁ信号の発生と位置取り込み完了との反応時間は極めて短くなります。最後の２軸(軸２と軸３)のみだけ位置のﾗｯﾁ機能が可能です。LTC3(軸２)とLTC4(軸３)は、ﾗｯﾁﾊﾟﾙｽを受信するのに使用されます。
_7443_set_ltc_logic()ｺﾏﾝﾄﾞでﾗｯﾁの論理を設定できます。
_7443_get_latch_data(AxisNo,CntNo,Pos)ｺﾏﾝﾄﾞでｶｳﾝﾀのﾗｯﾁﾃﾞｰﾀを獲得できます。２番目のﾊﾟﾗﾒｰﾀ ***CntNo*** は、ﾗｯﾁﾃﾞｰﾀが読まれるｶｳﾝﾀを示すのに使用されます。
また、***_7443_set_enable_inp()***ｺﾏﾝﾄﾞにより LTC ﾋﾟﾝ以外(LTC,ORG,ｶｳﾝﾀ 4,ｶｳﾝﾀ 5)でもﾗｯﾁすることができます。

◆ 関連ﾌｧﾝｸｼｮﾝ：
_7443_set_ltc_logic(),_7443_get_latch_data(), _7443_set_enable_inp()：セクション6.16参照

4.8 ﾊﾞｯｸﾗｯｼｭ補正と振動抑制

方向の変化があったときはいつも、PPCI7443 はｺﾏﾝﾄﾞを送る前にﾊﾞｯｸﾗｯｼｭ調整ﾊﾟﾙｽを出力します。***_7443_backlash_comp()***ｺﾏﾝﾄﾞは、ﾊﾞｯｸﾗｯｼｭ調整用のﾊﾟﾙｽ数を設定するのに使用されます。

ﾓｰﾀが停止するとき振動を最小にするために、PPCI7443 はｺﾏﾝﾄﾞ動作完了後すぐに、反対方向に 1 ﾊﾟﾙｽ出力し、そして同方向に 1 ﾊﾟﾙｽ出力することができます。***_7443_suppress_vibration()***ｺﾏﾝﾄﾞは T1 と T2 の設定に使用されます。（下記の図参照）



- 関連ﾌｧﾝｸｼｮﾝ：
  _7443_backlash_comp(),_7443_suppress_vibration()：セクション6.6 参照

4.9 ｿﾌﾄﾘﾐｯﾄ機能

PPCI7443 は各軸に２つのｿﾌﾄﾘﾐｯﾄを用意してあります。ｿﾌﾄﾘﾐｯﾄは、正しく設定されてるとき、物理的なﾘﾐｯﾄｽｲｯﾁとして動作するので、ﾕｰｻﾞｰの機械的ｼｽﾃﾑを保護するのに非常に便利です。
ｿﾌﾄﾘﾐｯﾄはｺﾝﾊﾟﾚｰﾀ１と２(ｾｸｼｮﾝ 4.7.1 参照)に組み込まれ、そして、比較源はｺﾏﾝﾄﾞﾎﾟｼﾞｼｮﾝｶｳﾝﾀです。動作理論はｺﾝﾊﾟﾚｰﾀ１と２にあらかじめ設定しておき、ｺﾏﾝﾄﾞﾎﾟｼﾞｼｮﾝｶｳﾝﾀがその設定値に達したとき、PPCI7443 が機械的ｴﾝﾄﾞﾘﾐｯﾄｽｲｯﾁが動作したかのように処理します。したがって、このﾌｧﾝｸｼｮﾝは、ﾊﾟﾙｽ出力を急停止するか減速停止します。

_7443_set_softlimit() :　　ｿﾌﾄﾘﾐｯﾄの設定
_7443_enable_softlimit() :　ｿﾌﾄﾘﾐｯﾄを有効に設定
_7443_diable_softlimit() :　ｿﾌﾄﾘﾐｯﾄを無効に設定

注意：ｿﾌﾄﾘﾐｯﾄは、ﾌｨｰﾄﾞﾊﾞｯｸﾎﾟｼﾞｼｮﾝ,ｺﾏﾝﾄﾞﾎﾟｼﾞｼｮﾝに適用されます。(ｾｸｼｮﾝ 4.4 参照)。ｿﾌﾄﾘﾐｯﾄ(***_7443_set_softlimit()***ｺﾏﾝﾄﾞ)を使用するときに動作比率が異なる場合、手動で対応するｺﾏﾝﾄﾞﾎﾟｼﾞｼｮﾝを計算する必要があります。

◆ 関連ﾌｧﾝｸｼｮﾝ：
_7443_set_softlimit(),_7443_enable_softlimit(), _7443_diable_softlimit()：セクション6.16参照

4.10　割り込み信号制御

PPCI7443 はﾎｽﾄ PC に INT 信号を発生させることができます。ｿﾌﾄｳｪｱ機能 ***_7443_int_control()***ｺﾏﾝﾄﾞのﾊﾟﾗﾒｰﾀ ***intFlag*** は割り込み動作を有効/無効の設定ができます。
割り込みが発生した後に、ﾌｧﾝｸｼｮﾝ***_7443_get_int_status()***ｺﾏﾝﾄﾞは、INT 信号についての情報を含む INT 状態を受けるのに使用されます。
PPCI7443 の INT 状態は２つの要素（error_int_status と event_int_status）で構成されます。event_int_status は通常の動作での割り込み要素で、それらは***_7443_set_int_factor()***ｺﾏﾝﾄﾞでﾏｽｸされます。error_int_status は PPCI7443 のエラー(ｱﾗｰﾑ)的な割り込み要素です。例えば EL、ALM･･･などで、これらの要素はﾏｽｸすることができません。下記の表は、これら２つの割り込み条件の定義です。

event_int_status:_7443_int_factor()を呼び出すことでﾏｽｸできる	
ﾋﾞｯﾄ	記述
0	通常の停止時
1	次のコマンドが継続スタート時
2	連続動作用ﾌﾟﾘﾚｼﾞｽﾀ 2 書き込み可能時
3	（予約）
4	加速開始時
5	加速終了時
6	減速開始時
7	減速終了時
8	（予約）
9	（予約）
10	脱調が発生時
11	汎用ｺﾝﾊﾟﾚｰﾀが比較時
12	軸０，軸１の比較ﾄﾘｶﾞｰ出力時
13	（予約）
14	軸２，軸３のﾗｯﾁ入力時
15	ＯＲＧが入力によるカウント値のラッチ時
16	ＳＤが入力ＯＮ時
17	（予約）
18	（予約）
19	同時スタート入力ＯＮ時
20~31	（予約）

Error_int_status:割り込み動作が ON していれば、ﾏｽｸできない	
ﾋﾞｯﾄ	記述
0	＋側ソフトリミットによる停止
1	－側ソフトリミットによる停止
2	（予約）
3	汎用ｺﾝﾊﾟﾚｰﾀが条件成立による停止
4	（予約）
5	＋側エンドリミットによる停止
6	－側エンドリミットによる停止
7	アラーム入力による停止
8	CSTP 入力による停止
9	CEMG 入力による停止
10	ＳＤ入力による停止
11	（予約）
12	補間データ異常による停止
13	補間動作中に他軸の異常停止による停止
14	ﾊﾟﾙｻｰ入力用ﾊﾞｯﾌｧｶｳﾝﾀｰのｵｰﾊﾞｰﾌﾛｰによる停止
15	補間動作時の位置決めｶｳﾝﾀのｶｳﾝﾄ範囲ｵｰﾊﾞｰによる停止
16	エンコーダ入力エラー（停止しません）
17	パルサー入力エラー（停止しません）
18~31	（予約）

◆ Events を Windows の下で割り込み信号を扱うのに使用
Windows の下で PPCI7443 からの割り込み信号を検出するために、ﾕｰｻﾞｰは最初に、Events 配列を作成しなければなりません。そして、割り込み信号を読み込むために PPCI7443 によって提供された機能を使用します。ｻﾝﾌﾟﾙﾌﾟﾛｸﾞﾗﾑは下記の通りです。

ステップ:

1. 割り込みｲﾍﾞﾝﾄを扱うためにｸﾞﾛｰﾊﾞﾙな値を定義する。それぞれのｲﾍﾞﾝﾄが一つの軸にﾘﾝｸされます。
   HANDLE hEvent[4];
2. 割り込みｲﾍﾞﾝﾄを有効にし、割り込み要素を設定、そして割り込みﾁｬﾝﾈﾙを有効にします。
   _7443_int_enable(0,hEvent);
   _7443_set_int_factor(0,0x01);　　//通常停止時発生

```
_7443_int_control(0,1);
```

3. 移動ｺﾏﾝﾄﾞで軸を移動開始します。
```
_7443_start_tr_move (0,12000,0,10000,0.1,0.1);
```

4. 軸０の割り込みを待ちます。
```
STS=WaitForSingleObject(hEvent[0],15000);
  ResetEvent(hEvent[0]);


  if( STS==WAIT_OBJECT_0 )
  {
      _7443_get_int_status(0, &error, &event);
      if( event == 0x01 ) …… ; // Success
  }
  else if( STS==WAIT_TIME_OUT)
  {
      // Time out, fail
  }
```

◆ DOS による PPCI7443 の割り込みｻｰﾋﾞｽﾙｰﾁﾝ(ISR)

ﾕｰｻﾞｰが DOS 環境でｱﾌﾟﾘｹｰｼｮﾝを開発する事が出来るように、DOS ﾌｧﾝｸｼｮﾝﾗｲﾌﾞﾗﾘが PPCI7443 に備えられています。このﾗｲﾌﾞﾗﾘもまた、ISR を扱ういくつかのﾌｧﾝｸｼｮﾝが用意されています。次の例を参考にして、ISR を扱うｱﾌﾟﾘｹｰｼｮﾝを作成して下さい。PCI バスには、複数の PPCI7443 が使用されているとき IRQ を共有をする機能があるので、各 PPCI7443 では、対応する ISR を必要とします。我々は ISR 名を下記のように直して、ﾗｲﾌﾞﾗﾘを使用するﾕｰｻﾞｰに提供します

_7422_isr0(void)、***_7422_isr1(void)***…etc.

ｻﾝﾌﾟﾙﾌﾟﾛｸﾞﾗﾑを下記に記述します。それは、２枚の PPCI7443 がｽﾛｯﾄに挿入されていると仮定して、１軸目と５軸目が ISR で共に動作するように求めると：

```
//ﾍｯﾀﾞｰﾌｧｲﾙの宣言
#include "pci_7443.h"

void main(void){
    I16 TotalCard,i;                    //ｶｰﾄﾞを初期化
    _7443_initial(&TotalCard);
    if(TotalCard==0)exit(1);
```

```
    _7443_set_int_factor(0,0x1);          //int 要素設定
    _7443_int_control(0,1);               //int 動作有効に設定
                ・
                ・                        //ﾒｲﾝにﾕｰｻﾞｰｺｰﾄﾞを挿入
                ・
                ・

    _7443_int_control(0,0);               //int 動作を無効に設定

    _7443_close();                        //PPCI7443 閉じる
}

void interrupt_7443_isr0(void){
    U16 irq_status;             //宣言
    U16 int_type;
    I16 i;
    U32 i_int_status1[4],i_int_status2[4];

disable();                                //すべての int 動作停止
    _7443_get_irq_status(0,&irq_status); //このｶｰﾄﾞ intﾁｪｯｸ
    if(irq_status){
          for(i=0;i<4;i++)_7443_enter_isr(i);     //isr に入る
          for(i=0;i<4;i++){
                _7443_get_int_type(I,&int_type);  //intﾀｲﾌﾟﾁｪｯｸ
                if(int_type & 0x1){
                    _7443_get_error_int(I,&int_status1[I]);

    //ｴﾗｰ INT にﾕｰｻﾞｰｺｰﾄﾞの挿入
    //
    //

          }
          if(int_type&0x2){
                _7443_get_event_int(I,&int_status2[I]);

    //ｲﾍﾞﾝﾄ INT にﾕｰｻﾞｰｺｰﾄﾞの挿入
    //
    //
          }
    }
    //card0 のすべての軸の終了
```

```
            for(i=0;i<4;i++)_7443_leave_isr(i);
        }
        else _7443_not_my_irq(0);

    //EOI を送る
        _OUTPORTB(0x20,0x20);
        _OUTPORTB(0xA0,0x20);
        enable();                           //int 動作可能
}

void interrupt_7443_isr1(void){}
void interrupt_7443_isr2(void){}
void interrupt_7443_isr3(void){}
void interrupt_7443_isr4(void){}
void interrupt_7443_isr5(void){}
void interrupt_7443_isr6(void){}
void interrupt_7443_isr7(void){}
void interrupt_7443_isr8(void){}
void interrupt_7443_isr9(void){}
void interrupt_7443_isra(void){}
void interrupt_7443_isrb(void){}
```

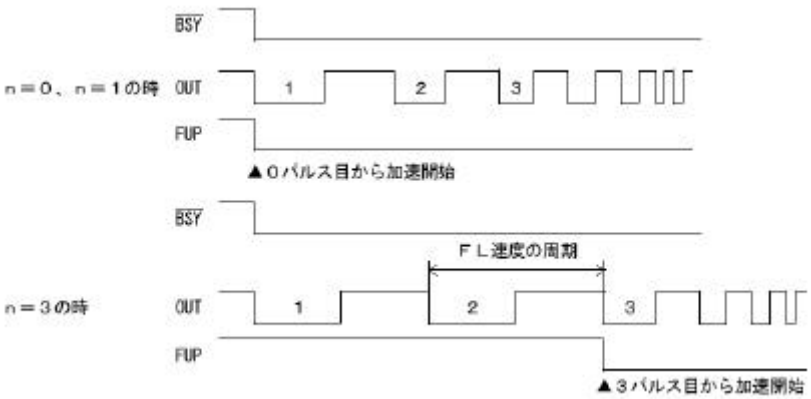
このｻﾝﾌﾟﾙﾌﾟﾛｸﾞﾗﾑによって、ﾕｰｻﾞｰは軸制御ｼｽﾃﾑで各軸からの割り込み信号を得ることができます。

- 関連ﾌｧﾝｸｼｮﾝ：
  _7443_int_control(),_7443_set_int_factor(),_7443_int_enable(), _7443_int_disable(),_7443_get_int_status(), _7443_link_interrupt(),_7443_get_int_type(),_7443_enter_isr(), _7443_leave_isr(),_7443_get_event_int(),_7443_get_error_int(), _7443_get_irq_status(),_7443_not_my_irq(),_7443_isr0~9,a,b：セクション6.14参照

4.11 ｱｲﾄﾞﾘﾝｸﾞ制御

加減速動作のｽﾀｰﾄ時に、開始速度(StrVer)で数ﾊﾟﾙｽｱｲﾄﾞﾘﾝｸﾞ出力してから加速を開始させることができます。
_7443_set_idle_pilse()ｺﾏﾝﾄﾞの ***idl_pulse***ﾊﾟﾗﾒｰﾀに設定したﾊﾟﾙｽ数だけ加速開始ﾀｲﾐﾝｸﾞを遅らせることができます。
位置決めﾓｰﾄﾞでこのﾌｧﾝｸｼｮﾝを使用した場合でも、全体の移動量は変化しません。

ｱｲﾄﾞﾘﾝｸﾞﾊﾟﾙｽの設定と加速開始ﾀｲﾐﾝｸﾞ



- 関連ﾌｧﾝｸｼｮﾝ：
  _7443_set_idle_pulse() : セクション6.6参照

5

PPCI7443 Utility

第２、第３章に従って基板を設定し、全てのﾊｰﾄﾞｳｪｱをｲﾝｽﾄｰﾙした後で、基板を動作させる前にﾁｪｯｸが必要です。本章では、制御ｼｽﾃﾑを確立するためにｶﾞｲﾄﾞﾗｲﾝを与え、そして正しく機能することを確かめるためにPPCI7443 を手動で動作させます。「PPCI7443 Utility」は PPCI7443 を使用する制御システムの設定、構成、試験、そしてﾃﾞﾊﾞｯｸﾞを行う簡単な方法を提供しています。

「PPCI7443 Utility」は画面解像度が 800 x 600 以上の Windows95/98 か Windows NT/2000/XP のみに利用可能であり、DOS で動作させることはできません。

5.1 PPCI7443 Utilityの起動

PPCI7443 のｿﾌﾄｳｪｱﾄﾞﾗｲﾊﾞ を Windows 95/98/NT/2000 にｲﾝｽﾄｰﾙした後に、「PPCI7443 Utility」ﾌﾟﾛｸﾞﾗﾑは<chosen path>/PPCI7443/Utility で見つけることができます。それを実行するには、ﾀﾞﾌﾞﾙｸﾘｯｸかﾃﾞｽｸﾄｯﾌﾟの start からﾌﾟﾛｸﾞﾗﾑ、そして PPCI7443 から、「PPCI7443 Utility」で実行できます。

5.2 PPCI7443 Utilityについて

PPCI7443 Utility を PPCI7443 で起動させる前に、下記の事項についてご注意願います。

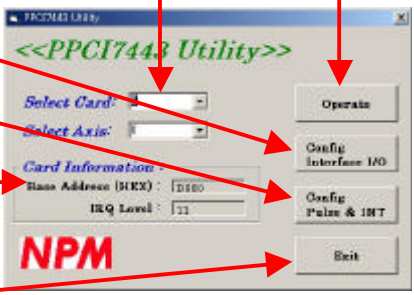
1. 「PPCI7443 Utility」は VB5.0 で書かれた Single Document Interface(SDI)ﾌﾟﾛｸﾞﾗﾑです。そして、画面解像度が 800x600 環境より高い Windows95/98 か Windows NT/2000/XP だけに利用可能で、Dos では動作できません。
2. 「PPCI7443 Utility」は PPCI7443 の設定もしくは構成をｾｰﾌﾞすることができます。そして、「PPCI7443 Utility」が再度起動されたとき、それらのｾｰﾌﾞ された構成が自動的に読み出されます。Windows ﾃﾞｨﾚｸﾄﾘの 7443.ini と 7443MC.ini の２つのﾌｧｲﾙはすべての設定と構成を保存するのに使用されます。
3. １個のｼｽﾃﾑから他のｼｽﾃﾑに構成をｺﾋﾟｰするのに、Windows ﾃﾞｨﾚｸﾄﾘの 7443.ini と 7443MC.ini をｺﾋﾟｰしてください
4. もしﾕｰｻﾞｰが「PPCI7443 Utility」によって設定された構成を使うならば、 DLL ﾌｧﾝｸｼｮﾝｺｰﾙ()が役立ちます。ﾕｰｻﾞｰのﾌﾟﾛｸﾞﾗﾑでこのﾌｧﾝｸｼｮﾝを呼び出した後で、ﾕｰｻﾞｰは「PPCI7443 Utility」によって設定された同じ構成で PPCI7443 を使用することができます。

5.3　PPCI7443 Utilityの画面

5.3.1 メイン画面

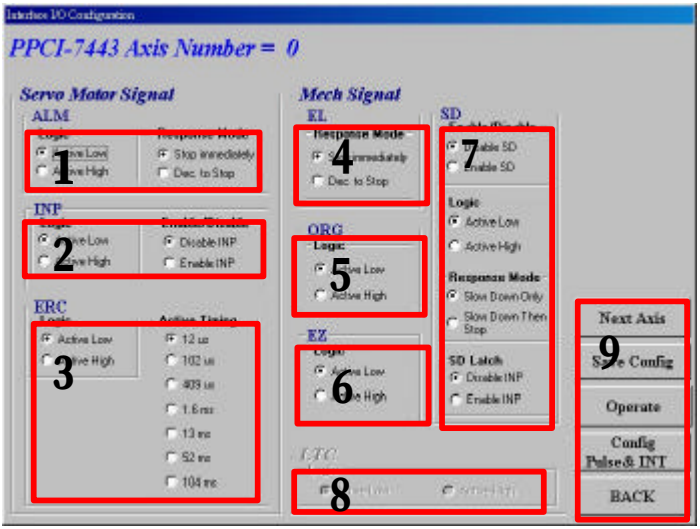
PPCI7443 が起動されるとﾒｲﾝ画面が表示されます。
下記の説明のように使用します。

- 軸と基板を選択
- 操作画面を表示 (ｾｸｼｮﾝ 5.3.4 を参照)
- 入出力ｲﾝﾀｰﾌｪｰｽ構成画面を表示(ｾｸｼｮﾝ 5.3.2 参照)
- ﾊﾟﾙｽ I/O と割り込み構成画面の表示(ｾｸｼｮﾝ 5.3.3 参照)
- 基板情報:
  関連機能
  _7443_get_base_addr(),
  _7443_get_irq_channel()
- PPCI7443 Utility の終了



5.3.2 入出力ｲﾝﾀｰﾌｪｰｽ構成画面

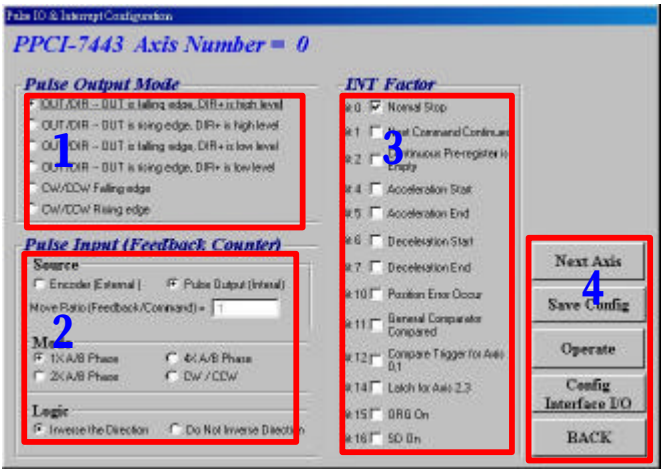
この画面でﾕｰｻﾞｰは、EL, ORG, EZ, ERC, ALM, INP, SD, および LTC の構成を設定できます。



1. **ALM Logic と Response mode：**
   ALM 信号の論理と処理ﾓｰﾄﾞを選択します。
   関連するﾌｧﾝｸｼｮﾝは***_7443_set_alm()***です。
2. **INP Logic と Enable/Disable：**
   INP 信号の論理と有効/無効を選択します。
   関連するﾌｧﾝｸｼｮﾝは***_7443_set_inp()***です。
3. **ERC Logic と Active timing：**
   ERC 信号の論理と動作時間を選択します。
   関連するﾌｧﾝｸｼｮﾝは***_7443_set_erc()***です。
4. **EL Response mode：**
   EL 信号の処理ﾓｰﾄﾞを選択します。
   関連するﾌｧﾝｸｼｮﾝは***_7443_set_el()***です。
5. **ORG Logic：**
   ORG 信号の論理を選択します。
   関連するﾌｧﾝｸｼｮﾝは***_7443_set_home_config()***です。
6. **EZ Logic：**
   EZ 信号の論理を選択します。
   関連するﾌｧﾝｸｼｮﾝは***_7443_set_home_config()***です。
7. **SD Configuration：**
   SD 信号の構成を設定します。
   関連するﾌｧﾝｸｼｮﾝは***_7443_set_sd()***です。
8. **LTC Logic：**
   LTC 信号の論理を選択します。
   関連するﾌｧﾝｸｼｮﾝは***_7443_set_ltc_logic()***です。
9. **押しボタン：**
   - **Next Axis:** このﾎﾞﾀﾝをｸﾘｯｸすると次の軸の設定操作ができます。
   - **Save Config：** このﾎﾞﾀﾝをｸﾘｯｸすると現在の設定ﾃﾞｰﾀを 7443.ini ﾌｧｲﾙに保存します。
   - **Operate:** 軸操作画面を表示します(ｾｸｼｮﾝ 5.3.4 参照)
   - **Config Pulse & INT:** ﾊﾟﾙｽ I/O と割り込み画面を表示します。(ｾｸｼｮﾝ 5.3.3 参照)
   - **Back:** このﾎﾞﾀﾝをｸﾘｯｸするとﾒｲﾝ画面に戻ります。

5.3.3 ﾊﾟﾙｽI/Oと割り込み構成画面

この画面では、ﾊﾟﾙｽ入力/出力、動作比率、および割り込み要素の構成が設定できます。

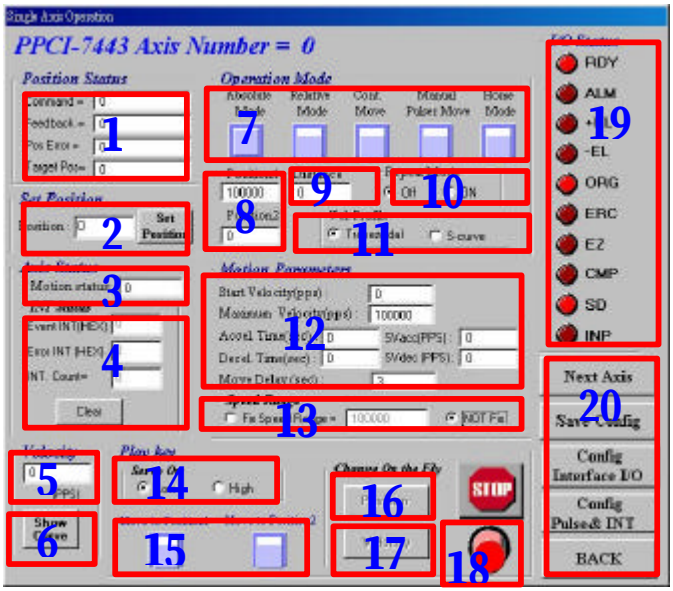


1. **Pulse Output Mode：**
   ﾊﾟﾙｽ信号(OUT,DIR)の出力ﾓｰﾄﾞを選択します。
   関連するﾌｧﾝｸｼｮﾝは***_7443_set_pls_outmode()***です。
2. **Pulse Input：**
   ﾊﾟﾙｽ入力信号(EA/EB)の構成を設定します。
   関連するﾌｧﾝｸｼｮﾝは***_7443_set_pls_intmode()***,
   _7443_set_feedback_src()です。
   Move Ratio: 現在動作させる目的の軸の比率(ﾌｨｰﾄﾞﾊﾞｯｸﾊﾟﾙｽ÷ｺﾏﾝﾄﾞﾊﾟﾙｽ)を設定します。この値は「0」であってはなりません。
   関連するﾌｧﾝｸｼｮﾝは***_7443_set_move_ratio()***です。
3. **INT Factor：**
   割り込みｲﾍﾞﾝﾄを起こす要素を選択します。
   関連するﾌｧﾝｸｼｮﾝは***_7443_set_int_factor()***です。
4. **押しボタン：**
   - **Next Axis:** このﾎﾞﾀﾝをｸﾘｯｸすると次の軸の設定操作ができます。
   - **Save Config：** このﾎﾞﾀﾝをｸﾘｯｸすると現在の設定ﾃﾞｰﾀを7443.ini ﾌｧｲﾙに保存します。
   - **Operate:** 軸操作画面を表示します(ｾｸｼｮﾝ 5.3.4 参照)

- **Config Interface I/O:** 入出力ｲﾝﾀｰﾌｪｰｽ構成画面を表示します。（ｾｸｼｮﾝ 5.3.2 参照）
- **Back:** このﾎﾞﾀﾝをｸﾘｯｸするとﾒｲﾝ画面に戻ります。

5.3.4 軸操作画面

この画面でﾕｰｻﾞｰは PPCI7443 が提供する 1 軸の動作機能（速度ﾓｰﾄﾞ動作、ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞ/ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞのﾌﾟﾘｾｯﾄ動作、手動ﾊﾟﾙｻｰ動作を含む）を学びそして操作することができます。



1. Position Reset:

- **Command:** ｺﾏﾝﾄﾞｶｳﾝﾀの値を表示します。
  関連するﾌｧﾝｸｼｮﾝは ***_7443_get_command()***です。
- **FeedBack:** ﾌｨｰﾄﾞﾊﾞｯｸｶｳﾝﾀの値を表示します。
  関連するﾌｧﾝｸｼｮﾝは ***_7443_get_position()***です。
- **Pos Error:** ﾎﾟｼﾞｼｮﾝｴﾗｰｶｳﾝﾀの値を表示します。
  関連するﾌｧﾝｸｼｮﾝは ***_7443_get_error_counter()***です。
- **Target Pos:** ﾀｰｹﾞｯﾄﾎﾟｼﾞｼｮﾝﾚｼﾞｽﾀの値を表示します。
  関連するﾌｧﾝｸｼｮﾝは ***_7443_get_error_counter()***です。

2. **Position Reset:**

   このﾎﾞﾀﾝをｸﾘｯｸするとすべてのﾎﾟｼﾞｼｮﾝｶｳﾝﾀの値を設定欄に入力した値に変更します。

   関連するﾌｧﾝｸｼｮﾝは、***_7443_set_Posision(), _7443_set_command(), _7443_reset_error_counter(), _7443_reset_target_pos()***です。

3. **Motion Status:**

   _7443_motion_done()ｺﾏﾝﾄﾞによる戻り値を表示します。

   関連するﾌｧﾝｸｼｮﾝは、***_7443_motion_done()***です。

4. **INT Status:**

   Event: **event_int_status** の値を HEX 値で表示します。
   関連するﾌｧﾝｸｼｮﾝは ***_7443_get_int_status()***です。

   Error: **error_int_status** の値を HEX 値で表示します。
   関連するﾌｧﾝｸｼｮﾝは ***_7443_get_int_status()***です。

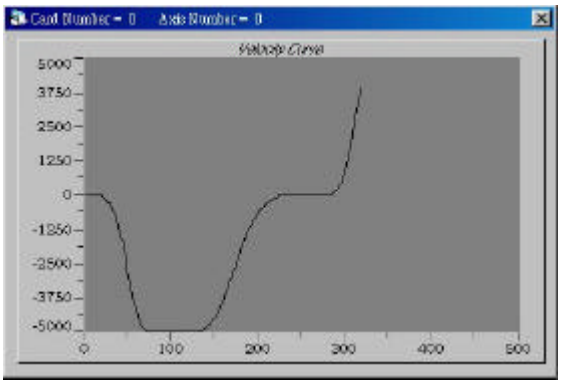
5. **Velocity:**

   移動速度の絶対値を pps の単位で表示します。

   関連するﾌｧﾝｸｼｮﾝは、***_7443_get_current_speed()***です。

6. **Show Velocity Curve Button:**

   このﾎﾞﾀﾝをｸﾘｯｸすると速度と時間を示すｸﾞﾗﾌを表示します。このｶｰﾌﾞは 100msec 毎に新しい速度ﾃﾞｰﾀを表示します。このﾎﾞﾀﾝを再度ｸﾘｯｸすると、この画面を閉じます。ｸﾞﾗﾌをｸﾘｯｸするとﾃﾞｰﾀがｸﾘｱされます。



7. **Operation Mode：** 操作ﾓｰﾄﾞを選択します。

   - **Absolute Mode：**

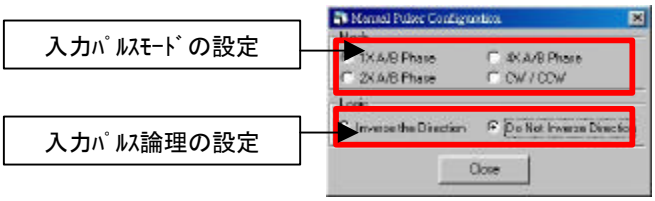
     軸の動作ﾓｰﾄﾞをｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞに設定します。移動の目標位置は Positon1 と Position2 を使用します。

     関連するﾌｧﾝｸｼｮﾝは
     _7443_start_ta_move(),_7443_start_sa_move()です。

● **Relative Mode：**
軸の動作ﾓｰﾄﾞをｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞに設定します。移動の目標位置は Distance を使用します。
関連するﾌｧﾝｸｼｮﾝは***_7443_start_tr_move(), _7443_start_sr_move()***です。

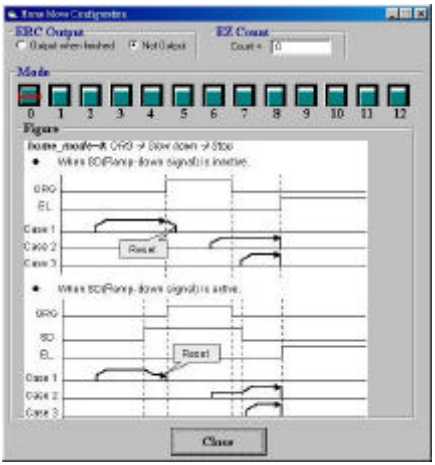
● **Cont. Move：**
軸の動作ﾓｰﾄﾞを一定速度の動作ﾓｰﾄﾞに設定します。
関連するﾌｧﾝｸｼｮﾝは***_7443_tv_move(),_7443_start_sv_move()***です。

● **Manual Pulser Move：**
軸の動作ﾓｰﾄﾞを手動のﾊﾟﾙｻｰﾓｰﾄﾞに設定します。このﾎﾞﾀﾝをｸﾘｯｸすると手動ﾊﾟﾙｻｰ構成画面が表示され、入力ﾊﾟﾙｽﾓｰﾄﾞ、ﾊﾟﾙｽ論理の設定が行えます。



● **Home Mode：**
軸の動作ﾓｰﾄﾞを原点復帰ﾓｰﾄﾞに設定します。このﾎﾞﾀﾝをｸﾘｯｸすると原点復帰構成画面が表示され、原点復帰ﾓｰﾄﾞが選択できます。
関連するﾌｧﾝｸｼｮﾝは ***_7443_set_home_config()***です。

ERC Output: 原点復帰完了時に ERC 信号を出力するかしないかを選択します。
EZ Count: 原点復帰ﾓｰﾄﾞで EZ 信号を使用するとき、EZ 信号をｶｳﾝﾄする数を設定します。
Mode: 原点復帰ﾓｰﾄﾞを選択します。
Figure: ここで示される図は個々の原点復帰ﾓｰﾄﾞの動作を説明します。
Close: このﾎﾞﾀﾝをｸﾘｯｸするとこの画面を閉じます。



8. Position：

ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞのときの目標位置を設定します。「Absolute Mode」が選択されているときのみ有効です。

9. Distance：

ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞのときの移動距離を設定します。「Relative Mode」が選択されているときのみ有効です。

10. Repeat Mode：

「ON」が選択されているとき、動作が繰り返しﾓｰﾄﾞになります(Forward ←→ Backward もしくは Position1 ←→ Position2)。「Absolute Mode」か「Relative Mode」のときのみ有効です。

11. Vel.Profile：

速度ﾓｰﾄﾞを選択します。「Trapezoidal」,「S-Curve」ともに「Absolute Mode」、「Relative Mode」、「Cont. Move」で選択可能です。

12. Motion Parameter：

１軸動作を実行するためのﾊﾟﾗﾒｰﾀを設定します。もし「Manual Pulser Move」が選択されていれば、速度と移動距離はﾊﾟﾙｽ入力によって決まるので、これらのﾊﾟﾗﾒｰﾀは無意味となります。

- **Start Velocity：**
  軸移動の開始速度を pps の単位で設定します。「Absolute Mode」か「Relative Mode」の場合は値だけが有効です。すなわち、-100 と 100 は同じです。「Cont. Move」では値と符号が有効になります。-100 は 100 のﾏｲﾅｽ方向を意味します。
- **Maximum Velocity：**
  軸移動の最高速度を pps の単位で設定します。「Absolute Mode」か「Relative Mode」の場合は値だけが有効です。すなわち、-5000 と 5000 は同じです。「Cont. Move」では値と符号が有効になります。-5000 は 5000 のﾏｲﾅｽ方向を意味します。
- **Accel.Time：**
  加速時間を秒の単位で設定します。
- **Decel.Time：**
  減速時間を秒の単位で設定します。
- **SVacc：**
  Ｓ字曲線動作の加速時間のＳ字領域を pps の単位で設定します。
- **SVdec：**
  Ｓ字曲線動作の減速時間のＳ字領域を pps の単位で設定します。
- **Move Delay：**
  「Repeat Mode」が「ON」に設定されているときのみ有効です。これは、PPCI7443 で繰り返し動作を行うときに、ここに設定した時間(秒)だけ遅らせて次の動作を実行します。

13. Speed Range：

軸移動の最大速度を設定します。もし「Not Fix」が選択されていれば、軸の最高速度は自動的に、ここに設定された速度に制限されます。（速度のｵｰﾊﾞｰﾗｲﾄﾞで越えることができない速度）

14. Servo On：

SVON 信号の出力状態を設定します。

関連するﾌｧﾝｸｼｮﾝは***_7443_set_servo()***です。

15. Play Key：

- **Left play button:**

  このﾎﾞﾀﾝをｸﾘｯｸすると前の設定に従ってﾊﾟﾙｽを出力します。

  Absolute Mode のとき、Position1 へ移動

  Relative Mode のとき、前方向へ移動

  Cont. Move のとき、速度設定に従って移動開始

  Manual Pulser Move のとき、軸をﾊﾟﾙｻｰ移動に入り込ませます。制限速度は「Maximum Velocity」によって設定された値です。

- **Right play button:**

  このﾎﾞﾀﾝをｸﾘｯｸすると前の設定に従ってﾊﾟﾙｽを出力します。

  Absolute Mode のとき、Position2 へ移動

  Relative Mode のとき、後ろ方向へ移動

  Cont. Move のとき、速度設定に従って移動開始

  Manual Pulser Move のとき、軸をﾊﾟﾙｻｰ移動に入り込ませます。制限速度は「Maximum Velocity」によって設定された値です。

16. Change Position On The Fly Button：

このﾎﾞﾀﾝが有効のとき現在動作の目標位置を変更することができます。新しい位置は「Position2」で定義します。「Position1」では定義できません。

関連するﾌｧﾝｸｼｮﾝは***_7443_p_change()***です。

17. Change Velocity On The Fly Button：

このﾎﾞﾀﾝが有効のとき現在動作の動作速度を変更することができます。新しい速度は「Maximum Velocity」で定義します。

関連するﾌｧﾝｸｼｮﾝは***_7443_v_change()***です。

18. Stop Button：

このﾎﾞﾀﾝをｸﾘｯｸすると PPCI7443 を減速停止させます。減速時間は「Decel.Time」で設定された時間です。

関連するﾌｧﾝｸｼｮﾝは***_7443_sd_stop()***です。

19. I/O Status：

軸入出力信号のｽﾃｰﾀｽです。LED 点灯で信号が ON を示し、消灯で信号が OFF を示します。

関連するﾌｧﾝｸｼｮﾝは ***_7443_get_io_status()***です。

20. 押しボタン：

- **Next Axis:** このﾎﾞﾀﾝをｸﾘｯｸすると次の軸の設定操作ができます。
- **Save Config：** このﾎﾞﾀﾝをｸﾘｯｸすると現在の設定ﾃﾞｰﾀを7443.ini ﾌｧｲﾙに保存します。
- **Config Interface I/O:** 入出力ｲﾝﾀｰﾌｪｰｽ構成画面を表示します。(ｾｸｼｮﾝ 5.3.2 参照)
- **Config Pulse & INT:** ﾊﾟﾙｽ I/O と割り込み画面を表示します。(ｾｸｼｮﾝ 5.3.3 参照)
- **Back:** このﾎﾞﾀﾝをｸﾘｯｸするとﾒｲﾝ画面に戻ります。

6

ファンクションライブラリ

本章は PPCI7443 のソフトウェアについて説明します。C のアプリケーションプログラム、Visual Basic または C++言語を開発するのにこれらのファンクションを使用することができます。もし、デルファイがプログラミング環境として用いられるのであれば、ヘッダーファイル 7443.h を手動で変換する必要があります。

6.1 ファンクションライブラリのリスト

初期設定 *セクション 6.3*

ファンクション名	記述
_7443_initial	ソフトウェアの初期化
_7443_close	ソフトウェアのクローズ
_7443_get_base_addr	PPCI7443 のベースアドレスの読み込み
_7443_get_irq_channel	PPCI7443 の IRQ 番号の読み込み
_7443_delay_time	指定時間のプログラムの遅延
_7443_config_from_file	構成ファイル(PPCI7443 Utility で作成された 7443.ini ファイル)に従って、PPCI7443 構成する
_7443_version_info	基板のハードウェアとソフトウェアのバージョンチェック

パルス入力/出力構成 *セクション 6.4*

ファンクション名	記述
_7443_set_pls_outmode	パルス出力モードの設定
_7443_set_pls_iptmode	エンコーダ入力モードの設定
_7443_set_feedback_src	カウンタ入力源の設定

一定速度モード

セクション 6.5

ﾌｧﾝｸｼｮﾝ名	記述
_7443_tv_move	台形動作動作で軸を一定速度まで加速
_7443_sv_move	Ｓ字曲線動作で軸を一定速度まで加速
_7443_v_change	速度のｵｰﾊﾞｰﾗｲﾄﾞ
_7443_sd_stop	減速停止
_7443_emg_stop	急停止
_7443_fix_speed_range	速度範囲の定義
_7443_unfix_speed_range	速度範囲制限の解除
_7443_get_current_speed	現在速度の読み込み
_7443_verify_speed	最高速度における加速時間の最小値と最大値のﾁｪｯｸ

１軸の位置決めモード

セクション 6.6

ﾌｧﾝｸｼｮﾝ名	記述
_7443_start_tr_move	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでの台形動作の実行
_7443_start_ta_move	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでの台形動作の実行
_7443_start_sr_move	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでのＳ字曲線動作の実行
_7443_start_sa_move	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでのＳ字曲線動作の実行
_7443_set_move_ratio	ｺﾏﾝﾄﾞﾊﾟﾙｽとﾌｨｰﾄﾞﾊﾞｯｸﾊﾟﾙｽの比率の設定
_7443_p_change	位置のｵｰﾊﾞｰﾗｲﾄﾞ
_7443_set_pcs_logic	PCS 信号の論理の設定(位置変更信号)
_7443_set_sd_pin	SD/PCS ﾋﾟﾝの設定
_7443_backlash_comp	ﾊﾞｯｸﾗｯｼｭ補正ﾊﾟﾙｽの設定
_7443_suppress_vibration	振動抑制時間の設定
_7443_set_idle_pulse	ｱｲﾄﾞﾘﾝｸﾞﾊﾟﾙｽの設定
_7443_delay_time	ﾀｲﾏｰﾓｰﾄﾞでの位置決め動作の実行
_7443_set_auto_rdp	ﾀﾞｳﾝﾎﾟｲﾝﾄの手動/自動設定

ﾌｧﾝｸｼｮﾝ名	記述
_7443_start_tr_move_xy	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでの台形動作によるＸとＹの２軸直線補間の実行
_7443_start_ta_move_xy	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでの台形動作によるＸとＹの２軸直線補間の実行
_7443_start_sr_move_xy	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでのＳ字曲線動作によるＸとＹの２軸直線補間の実行
_7443_start_sa_move_xy	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでのＳ字曲線動作によるＸとＹの２軸直線補間の実行
_7443_start_tr_move_zu	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでの台形動作によるＺとＵの２軸直線補間の実行
_7443_start_ta_move_zu	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでの台形動作によるＺとＵの２軸直線補間の実行
_7443_start_sr_move_zu	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでのＳ字曲線動作によるＺとＵの２軸直線補間の実行
_7443_start_sa_move_zu	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでのＳ字曲線動作によるＺとＵの２軸直線補間の実行
_7443_start_tr_line2	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでの台形動作による任意の２軸直線補間の実行
_7443_start_sr_line2	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでのＳ字曲線動作による任意の２軸直線補間の実行
_7443_start_ta_line2	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでの台形動作による任意の２軸直線補間の実行
_7443_start_sa_line2	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでのＳ字曲線動作による任意の２軸直線補間の実行
_7443_start_tr_line3	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでの台形動作による任意の３軸直線補間の実行
_7443_start_sr_line3	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでのＳ字曲線動作による任意の３軸直線補間の実行
_7443_start_ta_line3	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでの台形動作による任意の３軸直線補間の実行
_7443_start_sa_line3	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでのＳ字曲線動作による任意の３軸直線補間の実行
_7443_start_tr_line4	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでの台形動作による４軸直線補間の実行
_7443_start_sr_line4	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでのＳ字曲線動作による４軸直線補間の実行
_7443_start_ta_line4	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでの台形動作による４軸直線補間の実行
_7443_start_sa_line4	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでのＳ字曲線動作による４軸直線補間の実行
_7443_set_line_move_mode	連続直線補間ﾓｰﾄﾞの設定
_7443_set_axis_option	補間速度ﾓｰﾄﾞの選択

円弧補間動作

セクション 6.8

ﾌｧﾝｸｼｮﾝ名	記述
_7443_start_a_arc_xy	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでのＸとＹの円弧補間の実行
_7443_start_r_arc_xy	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでのＸとＹの円弧補間の実行
_7443_start_a_arc_zu	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでのＺとＵの円弧補間の実行
_7443_start_r_arc_zu	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでのＺとＵの円弧補間の実行
_7443_start_a_arc2	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでの任意の２軸円弧補間の実行
_7443_start_r_arc2	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでの任意の２軸円弧補間の実行
_7443_start_ta_arc_xy	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでの台形動作によるＸとＹの円弧補間の実行
_7443_start_sa_arc_xy	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでのＳ字曲線動作によるＸとＹの円弧補間の実行
_7443_start_tr_arc_xy	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでの台形動作によるＸとＹの円弧補間の実行
_7443_start_sr_arc_xy	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでのＳ字曲線動作によるＸとＹの円弧補間の実行
_7443_start_ta_arc_zu	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでの台形動作によるＺとＵの円弧補間の実行
_7443_start_sa_arc_zu	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでのＳ字曲線動作によるＺとＵの円弧補間の実行
_7443_start_tr_arc_zu	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでの台形動作によるＺとＵの円弧補間の実行
_7443_start_sr_arc_zu	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでのＳ字曲線動作によるＺとＵの円弧補間の実行
_7443_start_ta_arc2	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでの台形動作による任意の２軸円弧補間の実行
_7443_start_sa_arc2	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでのＳ字曲線動作による任意の２軸円弧補間の実行
_7443_start_tr_arc2	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでの台形動作による任意の２軸円弧補間の実行
_7443_start_sr_arc2	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでのＳ字曲線動作による任意の２軸円弧補間の実行

原点復帰

セクション 6.9

ﾌｧﾝｸｼｮﾝ名	記述
_7443_set_home_config	原点復帰ﾓｰﾄﾞ、論理等の設定
_7443_home_move	原点復帰動作の実行
_7443_escape_home	原点抜け出し動作の実行
_7443_home_search	原点ｻｰﾁ動作の実行（原点ｵﾌｾｯﾄなし）
_7443_auto_home_search	原点ｻｰﾁ動作の実行（原点ｵﾌｾｯﾄあり）

パルサー動作

セクション 6.10

ﾌｧﾝｸｼｮﾝ名	記述
_7443_set_pulser_iptmode	ﾊﾟﾙｻｰ入力ﾓｰﾄﾞの設定
_7443_pulser_vmove	ﾊﾟﾙｻｰ速度移動の開始
_7443_pulser_pmove	ﾊﾟﾙｻｰ位置移動の開始
_7443_pulser_home_move	ﾊﾟﾙｻｰ原点復帰動作の開始
_7443_pulser_r_line2	ﾊﾟﾙｻｰによる任意２軸直線補間の開始
_7443_pulser_r_arc2	ﾊﾟﾙｻｰによる任意２軸円弧補間の開始

軸の動作状態

セクション 6.11

ﾌｧﾝｸｼｮﾝ名	記述
_7443_motion_done	軸の状態の読み込み

ｻｰﾎﾞﾄﾞﾗｲﾊﾞｲﾝﾀｰﾌｪｰｽ

セクション 6.12

ﾌｧﾝｸｼｮﾝ名	記述
_7443_set_alm	ｱﾗｰﾑ論理とｱﾗｰﾑﾓｰﾄﾞの設定
_7443_set_inp	INP 論理と INP ﾓｰﾄﾞの設定
_7443_set_erc	ERC 論理と出力時間の設定
_7443_set_servo	SVON 信号(汎用出力)の ON/OFF
_7443_set_sd	SD 論理と SD ﾓｰﾄﾞの設定
_7443_set_el	EL 論理と EL ﾓｰﾄﾞの設定

入出力モニター

セクション 6.13

ﾌｧﾝｸｼｮﾝ名	記述
_7443_get_io_status	PPCI7443 の I/O 状態の読み込み

割り込み信号制御

セクション 6.14

ﾌｧﾝｸｼｮﾝ名	記述	
_7443_int_control	INT 動作の有効/無効の設定	
_7443_int_enable	割り込みｲﾍﾞﾝﾄを有効に設定	(Windows のみ)
_7443_int_disable	割り込みｲﾍﾞﾝﾄを無効に設定	(Windows のみ)
_7443_get_int_status	INT 状態の読み込み	(Windows のみ)
_7443_link_interrupt	割り込みｺｰﾙﾊﾞｯｸﾌｧﾝｸｼｮﾝのﾘﾝｸ設定	(Windows のみ)
_7443_set_int_factor	INT 要素の設定	
_7443_get_int_type	INT ﾀｲﾌﾟの読み込み	(DOS のみ)
_7443_enter_isr	割り込みｻｰﾋﾞｽﾙｰﾁﾝに入る	(DOS のみ)
_7443_leave_isr	割り込みｻｰﾋﾞｽﾙｰﾁﾝから出る	(DOS のみ)
_7443_get_event_int	割り込みｲﾍﾞﾝﾄ状態の読み込み	(DOS のみ)
_7443_get_error_int	割り込みｴﾗｰ状態の読み込み	(DOS のみ)
_7443_get_irq_status	IRQ 状態の読み込み	(DOS のみ)
_7443_not_my_irq	自分の IRQ でない	(DOS のみ)
_7443_isr0~9、a、b	割り込みｻｰﾋﾞｽﾙｰﾁﾝ	(DOS のみ)
_7443_set_axis_stop_int	軸の割り込み停止を有効に設定	
_7443_mask_axis_stop_int	軸の割り込み停止をマスク	

位置の制御とカウンタ

セクション 6.15

ﾌｧﾝｸｼｮﾝ名	記述
_7443_get_position	ﾌｨｰﾄﾞﾊﾞｯｸﾎﾟｼﾞｼｮﾝｶｳﾝﾀの値の読み込み
_7443_set_position	ﾌｨｰﾄﾞﾊﾞｯｸﾎﾟｼﾞｼｮﾝｶｳﾝﾀの設定
_7443_get_command	ｺﾏﾝﾄﾞﾎﾟｼﾞｼｮﾝｶｳﾝﾀの値の読み込み
_7443_set_command	ｺﾏﾝﾄﾞﾎﾟｼﾞｼｮﾝｶｳﾝﾀの設定
_7443_get_error_counter	ﾎﾟｼﾞｼｮﾝｴﾗｰｶｳﾝﾀの値の読み込み
_7443_reset_error_counter	ﾎﾟｼﾞｼｮﾝｴﾗｰｶｳﾝﾀの値のﾘｾｯﾄ
_7443_get_general_counter	汎用ｶｳﾝﾀの値の読み込み
_7443_set_general_counter	汎用ｶｳﾝﾀの設定
_7443_get_target_pos	ﾀｰｹﾞｯﾄﾎﾟｼﾞｼｮﾝﾚｺｰﾀﾞの値の読み込み
_7443_reset_target_pos	ﾀｰｹﾞｯﾄﾎﾟｼﾞｼｮﾝﾚｺｰﾀﾞの値のﾘｾｯﾄ
_7443_get_rest_command	動作終了までの残りﾊﾟﾙｽの読み込み
_7443_check_rdp	ｽﾛｰﾀﾞｳﾝﾎﾟｲﾝﾄのﾃﾞｰﾀﾁｪｯｸ

位置の比較とラッチ

セクション 6.16

ﾌｧﾝｸｼｮﾝ名	記述
_7443_set_ltc_logic	LTC 信号の論理の設定
_7443_get_latch_data	ﾗｯﾁｶｳﾝﾀﾃﾞｰﾀの読み込み
_7443_set_soft_limit	ｿﾌﾄﾘﾐｯﾄの設定
_7443_enable_soft_limit	ｿﾌﾄﾘﾐｯﾄ機能を有効に設定
_7443_disable_soft_limit	ｿﾌﾄﾘﾐｯﾄ機能を無効に設定
_7443_set_error_counter_check	失ｽﾃｯﾌﾟの検出
_7443_set_general_comparator	汎用ｺﾝﾊﾟﾚｰﾀの設定
_7443_set_trigger_comparator	ﾄﾘｶﾞｰｺﾝﾊﾟﾚｰﾀの設定
_7443_set_trigger_type	ﾄﾘｶﾞｰの出力ﾀｲﾌﾟの設定
_7443_check_compare_data	現在のｺﾝﾊﾟﾚｰﾀﾃﾞｰﾀのﾁｪｯｸ
_7443_check_compare_status	現在のｺﾝﾊﾟﾚｰﾀ状態のﾁｪｯｸ
_7443_set_auto_compare	自動読み込みのためのｺﾝﾍﾟｱﾃﾞｰﾀの設定
_7443_build_compare_function	一定間隔でｺﾝﾍﾟｱﾃﾞｰﾀを作る
_7443_build_compare_table	ｺﾝﾍﾟｱﾃｰﾌﾞﾙでｺﾝﾍﾟｱﾃﾞｰﾀを作る
_7443_cmp_change	ｺﾝﾊﾟﾚｰﾀにによる速度変化
_7443_set_enable_inp	ｶｳﾝﾀをﾗｯﾁする信号の選択

連続動作

セクション 6.17

ﾌｧﾝｸｼｮﾝ名	記述
_7443_set_continuous_move	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでの連続動作の有効/無効設定
_7443_check_continuous_buffer	連続動作用ﾌﾟﾘﾚｼﾞｽﾀの空きﾁｪｯｸ

複数軸の同時操作

セクション 6.18

ﾌｧﾝｸｼｮﾝ名	記述
_7443_set_tr_move_all	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでの台形動作による同時ｽﾀｰﾄ設定
_7443_set_ta_move_all	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでの台形動作による同時ｽﾀｰﾄ設定
_7443_set_sr_move_all	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでのＳ字曲線動作による同時ｽﾀｰﾄ設定
_7443_set_sa_move_all	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでのＳ字曲線動作による同時ｽﾀｰﾄ設定
_7443_start_move_all	同時ｽﾀｰﾄの実行
_7443_stop_move_all	同時ｽﾄｯﾌﾟの実行
_7443_set_sync_option	同期ｽﾀｰﾄﾓｰﾄﾞの設定
_7443_set_sync_stop_mode	CSTOP 信号が入力された時の停止ﾓｰﾄﾞの設定

汎用ＴＴＬ出力

セクション 6.19

ﾌｧﾝｸｼｮﾝ名	記述
_7443_output	ﾃﾞｼﾞﾀﾙ出力の ON/OFF
_7443_get_dio_status	ﾃﾞｼﾞﾀﾙ出力状態の読み込み

6.2 C/C++ﾌﾟﾛｸﾞﾗﾐﾝｸﾞﾗｲﾌﾞﾗﾘ

このセクションではすべてのﾌｧﾝｸｼｮﾝの詳細を述べます。ｺﾏﾝﾄﾞの原型といくつかの共通のﾃﾞｰﾀﾀｲﾌﾟはPPCI7443.Hで収縮されます。これらのﾃﾞｰﾀﾀｲﾌﾟはPPCI7443ﾗｲﾌﾞﾗﾘによって使用されます。ｱﾌﾟﾘｹｰｼｮﾝﾌﾟﾛｸﾞﾗﾑでこれらのﾃﾞｰﾀﾀｲﾌﾟを使うことを勧めます。下記の表はﾃﾞｰﾀﾀｲﾌﾟ名とそれらの範囲を示します。

ﾀｲﾌﾟ名	記述	範囲
U8	8ビットのASCIIキャラクタ	0～255
I16	16ビットの符号付き整数	-32768～32767
U16	16ビットの符号無し整数	0～65535
I32	32ビットの符号付きの長い整数	-2147483648～2147483647
U32	32ビットの符号無しの長い整数	0～4294967295
F32	32ビットの単精度浮動小数点	-3.402823E38～3.402823E38
F64	64ビットの倍精度浮動小数点	-1.797683134862315E309 ～ 1.797683134862315E308
Boolean	ブール論理値	TRUE,FALSE

PPCI7443のｿﾌﾄｳｪｱﾄﾞﾗｲﾊﾞのﾌｧﾝｸｼｮﾝは、機能の本当の意味を表すのにﾌﾙﾈｰﾑを使用しています。ネーミング規定規則はは下記の通りである。

'C'ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ環境で:

_{hardware_model} _{action_name}.e.g._7443_Initial().

CとVBﾌｧﾝｸｼｮﾝの間の違いを認識するために、大文字の「B」をそのﾌｧﾝｸｼｮﾝの先頭において表しました。例えば、B_7443_Initial()はVisual Basic.で記述するﾌｧﾝｸｼｮﾝを意味します。

6.3 初期設定

@名前

_7443_Initial	PPCI7443のｿﾌﾄｳｪｱ初期設定
_7443_close	PPCI7443に関するｿﾌﾄｳｪｱのｸﾛｰｽﾞ
_7443_get_base_addr	PPCI7443のﾍﾞｰｽｱﾄﾞﾚｽの取得
_7443_get_irq_channel	PPCI7443のIRQ番号の取得
_7443_delay_time	指定時間次の実行を遅らせる
_7443_config_from_file	構成ﾌｧｲﾙ「7443.ini」に従って、PPCI7443を構成する
_7443_version_info	基板のﾊﾞｰｼﾞｮﾝ情報の取得

@記述

_7443_Initial

PPCI7443 を初期化するのに使用します。すべての PPCI7443 は他のﾌｧﾝｸｼｮﾝを呼び出す前に、このﾌｧﾝｸｼｮﾝによって初期化されなければなりません。

_7443_close

PPCI7443 を閉じて、ｱﾌﾟﾘｹｰｼｮﾝ終了時に PPCI7443 の関連したﾘｿｰｽをﾘﾘｰｽするのに使用します。

_7443_get_irq_channel

PPCI7443 の IRQ 番号を取得するのに使用します。

_7443_get_base_addr

PPCI7443 のﾍﾞｰｽｱﾄﾞﾚｽを取得するのに使用します。

_7443_delay_time

この機能はプログラムを指定時間(msec)遅らせて実行するのに使用します。

_7443_config_from_file:

PPCI7443 の各信号の設定を、「PPCI7443 Utility」で設定された構成ﾌｧｲﾙ(PPCI7443.ini)にするときに使用します。PPCI7443.ini ﾌｧｲﾙは「PPCI7443 Utility」で設定した構成を「save config」ﾎﾞﾀﾝを押すことでﾌｧｲﾙに保存されます。ﾊﾟﾗﾒｰﾀで PPCI7443.ini ﾌｧｲﾙを指定することで、各信号の構成が自動的に「PPCI7443 Utility」で設定した構成と同じになります。

このﾌｧﾝｸｼｮﾝが実行されると、下記のﾌｧﾝｸｼｮﾝが呼び出されたと同じように 7443.ini ﾌｧｲﾙに保存されているﾃﾞｰﾀにより、PPCI7443 は構成されます。

- **_7443_set_pls_outmode**
- **_7443_set_feedback_src**
- **_7443_set_pls_iptmode**

- **_7443_set_home_config**
- **_7443_set_int_factor**
- **_7443_set_el**
- **_7443_set_ltc_logic**
- **_7443_set_erc**
- **_7443_set_sd**
- **_7443_set_alm**
- **_7443_set_inp**
- **_7443_set_move_ratio**

_7443_version_info

PPCI7443 のﾊｰﾄﾞｳｪｱ、ｿﾌﾄｳｪｱのﾊﾞｰｼﾞｮﾝ情報を取得するのに使用します。

@構文

C/C++(DOS、Windows95/NT)

```
I16 _7443_initial(I16 *existCards);
I16 _7443_close(void);
I16 _7443_get_irq_channel(I16 cardNo, U16 *irq_no );
I16 _7443_get_base_addr(I16 cardNo, U16 *base_addr );
I16 _7443_delay_time(I16 AxisNo, Double MiniSec);
I16 _7443_config_from_file(char *filename);
I16 _7443_version_info(I16 CardNo, U16 *HardwareInfo, U16 *SoftwareInfo,
                       U16 *DriverInfo);
```

Visual Basic(Windows95/NT)

```
B_7443_initial (existCards As Integer) As Integer
B_7443_close () As Integer
B_7443_get_irq_channel (ByVal CardNo As Integer, irq_no As Integer) As
                       Integer
B_7443_get_base_addr (ByVal CardNo As Integer, base_addr As Integer)
                       As Integer
B_7443_delay_time (ByVal AxisNo As Integer, ByVal MiniSec As Double) As
                       Integer
B_7443_config_from_file(ByVal filename As string)as integer
B_7443_version_info (ByVal CardNo As Integer, HardwareInfo As Integer,
                       SoftwareInfo As Integer, DriverInfo As Integer) As
                       Integer
```

@パラメータ

***existCards:**	PPCI7443 の枚数
cardNo:	基板番号
***irq_no:**	指定された PPCI7443 の IRQ 番号
***base_addr:**	指定された PPCI7443 のベースアドレス
***filename:**	PPCI7443 の構成を記録する指定されたファイル名 このﾌｧｲﾙは PPCI7443 の「PPCI7443 Utility」によって作成
AxisNor:	軸番号
MiniSec:	遅延時間（単位は msec）
***Hardwareinfo:**	ﾊｰﾄﾞｳｪｱﾊﾞｰｼﾞｮﾝの読み込み値
***Softwareinfo:**	ｿﾌﾄｳｪｱﾗｲﾌﾞﾗﾘﾊﾞｰｼﾞｮﾝの読み込み値
***Driverinfo:**	ﾃﾞﾊﾞｲｽﾄﾞﾗｲﾊﾞｰﾊﾞｰｼﾞｮﾝの読み込み値

@復帰コード

ERR_NoError	: 0
ERR_NoCardFound	: 24
ERR_PCIBiosNotExist	: 8
ERR_ConigFileOpenError	: 39

6.4 ﾊﾟﾙｽ入力/出力構成

@名前

_7443_set_pls_outmode ﾊﾟﾙｽ出力ﾓｰﾄﾞの設定
_7443_set_pls_iptmode ｴﾝｺｰﾀﾞ入力ﾓｰﾄﾞの設定
_7443_set_feedback_src ｶｳﾝﾀ入力源の設定

@記述

_7443_set_pls_outmode

ｺﾏﾝﾄﾞﾊﾟﾙｽの出力ﾓｰﾄﾞを設定します。ｺﾏﾝﾄﾞﾊﾟﾙｽ出力には６個のﾓｰﾄﾞがあります。

_7443_set_pls_iptmode

外部からのﾌｨｰﾄﾞﾊﾞｯｸﾊﾟﾙｽの入力ﾓｰﾄﾞを設定します。ﾌｨｰﾄﾞﾊﾞｯｸﾊﾟﾙｽ入力には４つのﾀｲﾌﾟがあります。

このﾌｧﾝｸｼｮﾝは***_7443_set_feedback_src()***ｺﾏﾝﾄﾞのﾊﾟﾗﾒｰﾀ ***Src*** が有効に設定されているときだけ有効です。

_7443_set_feedback_src

外部からのｴﾝｺｰﾀﾞﾌｨｰﾄﾞﾊﾞｯｸﾊﾟﾙｽをこのｼｽﾃﾑで使用するならば、***Src***ﾊﾟﾗﾒｰﾀを「0」に設定します。その時、内部の 28 ﾋﾞｯﾄｱｯﾌﾟ/ﾀﾞｳﾝｶｳﾝﾀは ***_7443_set_pls_iptmode()***ｺﾏﾝﾄﾞで設定されたﾓｰﾄﾞでｶｳﾝﾄします。***Src***ﾊﾟﾗﾒｰﾀを「1」に設定するとｶｳﾝﾀはｺﾏﾝﾄﾞﾊﾟﾙｽを数えます。

@構文

C/C++(DOS、Windows95/NT)

```
I16 _7443_set_pls_outmode(I16 AxisNo, I16 pls_outmode);
I16 _7443_set_pls_iptmode(I16 AxisNo, I16 pls_iptmode, I16 pls_logic);
I16 _7443_set_feedback_src(I16 AxisNo, I16 Src);
```

Visual Basic(Windows95/NT)

```
B_7443_set_pls_outmode (ByVal AxisNo As Integer, ByVal pls_outmode As
                        Integer) As Integer
B_7443_set_pls_iptmode (ByVal AxisNo As Integer, ByVal pls_iptmode As
                        Integer, ByVal pls_logic As Integer) As Integer
B_7443_set_feedback_src (ByVal AxisNo As Integer, ByVal Src As Integer)
                        As Integer
```

@パラメータ

AxisNo:	軸番号
pls_outmode:	OUT,DIR ﾋﾟﾝからのｺﾏﾝﾄﾞﾊﾟﾙｽのﾓｰﾄﾞ
pls_outmode=0：	OUT/DIR OUT：立ち下がりｴｯｼﾞ，DIR+：は High
pls_outmode=1：	OUT/DIR OUT：立ち上がりｴｯｼﾞ，DIR+：は High
pls_outmode=2：	OUT/DIR OUT：立ち下がりｴｯｼﾞ，DIR+：は Low
pls_outmode=3：	OUT/DIR OUT：立ち上がりｴｯｼﾞ，DIR+：は Low
pls_outmode=4：	CW /CCW　　立ち下がりｴｯｼﾞ
pls_outmode=5：	CW /CCW　　立ち上がりｴｯｼﾞ
pls_iptmode:	EA,EB のｴﾝｺｰﾀﾞﾌｨｰﾄﾞﾊﾞｯｸﾊﾟﾙｽのﾓｰﾄﾞ
pls_iptmode=0：	１逓倍 AB 相ﾓｰﾄﾞ
pls_iptmode=1：	２逓倍 AB 相ﾓｰﾄﾞ
pls_iptmode=2：	４逓倍 AB 相ﾓｰﾄﾞ
pls_iptmode=3：	CW/CCW ﾊﾟﾙｽ入力ﾓｰﾄﾞ
pls_logic:	ｴﾝｺｰﾀﾞﾌｨｰﾄﾞﾊﾞｯｸﾊﾟﾙｽの論理
pls_logic=0：	Normal Low
pls_logic=1：	Normal High
Src:	ｶｳﾝﾀ入力源
Src=0：	外部からのﾌｨｰﾄﾞﾊﾞｯｸﾊﾟﾙｽ
Src=1：	ｺﾏﾝﾄﾞﾊﾟﾙｽ

@復帰コード

ERR_NoError　　：0

6.5　速度モード動き

@名前

_7443_tv_move	台形動作で軸を定速度まで加速
_7443_sv_move	Ｓ字曲線動作で軸を定速度まで加速
_7443_v_change	速度のｵｰﾊﾞｰﾗｲﾄﾞ機能
_7443_sd_stop	減速停止
_7443_emg_stop	急停止
_7443_fix_speed_range	速度範囲の定義
_7443_unfix_speed_range	速度範囲が制限の解除
_7443_get_current_speed	現在速度の読み込み
_7443_verify_speed	最高速度における加速時間の最大値と最小値のﾁｪｯｸ

@記述

_7443_tv_move

台形動作で軸を指定された定速度に加速するのに使用します。軸は速度が変えられるか停止命令されるまで一定速度で移動し続けます。速度ﾊﾟﾗﾒｰﾀの符号が動作方向を指示します。

_7443_sv_move

Ｓ字曲線動作で軸を指定された定速度に加速するのに使用します。軸は速度が変えられるか停止命令されるまで一定速度で移動し続けます。速度ﾊﾟﾗﾒｰﾀの符号が動作方向を指示します。

_7443_v_change

台形動作かＳ字曲線動作で移動速度を変更します。このﾌｧﾝｸｼｮﾝを実行する前に、***_7443_fix_speed_range()***ｺﾏﾝﾄﾞで速度範囲を定義する必要があります。また、***_7443_v_change()***ｺﾏﾝﾄﾞはﾌﾟﾘｾｯﾄﾓｰﾄﾞのときでも同様に適応可能です。

注意:速度動作は最初の動作ﾓｰﾄﾞ（Ｓ字曲線、台形）によって決められます。Ｓ字曲線で使用するとき、軸の動作を純粋なＳ字曲線に設定して下さい。このﾌｧﾝｸｼｮﾝにはいくつかの制限があるので、使用前にｾｸｼｮﾝ 4.6.1 をご参照下さい。

_7443_sd_stop

軸を停止させるのに台形動作かＳ字曲線動作で減速して止めるのに使用します。ﾌﾟﾘｾｯﾄ移動（台形、Ｓ字曲線）、手動動作、原点復帰動作が実行されているときに有効です。

注意：速度動作は最初の動作ﾓｰﾄﾞによって決められます。

_7443_emg_stop

すぐに軸を止めるのに使用します。ﾌﾟﾘｾｯﾄ移動（台形、Ｓ字曲線）、手動動作、原点復帰動作が実行されているときに有効です。

_7443_fix_speed_range

速度範囲を定義するのに使用します。軸の変化速度を抑制する場合、このﾌｧﾝｸｼｮﾝを軸移動開始前に呼び出さなければなりません。

_7443_unfix_speed_range

速度範囲の抑制ﾌｧﾝｸｼｮﾝを解除するのに使用します。

_7443_get_current_speed

指定された軸の現在のﾊﾟﾙｽ出力率（出力周波数）を読み込むのに使用します。それはどんな操作ﾓｰﾄﾞのときでも操作可能です。

_7443_verify_speed

速度軌跡において加速時間の最小及び最大設定時間がいくつであるか調べる機能です。

@構文

C/C++(DOS、Windows95/NT)

```
I16 _7443_tv_move(I16 AxisNo, F64 StrVel, F64 MaxVel, F64 Tacc);
I16 _7443_sv_move(I16 AxisNo, F64 StrVel, F64 MaxVel, F64 Tacc, F64
                  SVacc);
I16 _7443_v_change(I16 AxisNo, F64 NewVel, F64 Tacc);
I16 _7443_sd_stop(I16 AxisNo,F64 Tdec);
I16 _7443_emg_stop(I16 AxisNo);
F64 _7443_fix_speed_range(I16 AxisNo, F64 MaxVel);
I16 _7443_unfix_speed_range(I16 AxisNo);
I16 _7443_get_current_speed(I16 AxisNo, F64 *speed);
F64 _7443_verify_speed(F64 StrVel,F64 MaxVel,F64 *minAccT,F64
                  *maxAccT, F64 MaxSpeed);
```

Visual Basic(Windows95/NT)

```
B_7443_tv_move (ByVal AxisNo As Integer, ByVal StrVel As Double, ByVal
                  MaxVel As Double, ByVal Tacc As Double) As
                  Integer
B_7443_sv_move (ByVal AxisNo As Integer, ByVal StrVel As Double, ByVal
                  MaxVel As Double, ByVal Tacc As Double, ByVal
                  SVacc As Double) As Integer
B_7443_v_change (ByVal AxisNo As Integer, ByVal NewVel As Double,
                  ByVal TimeSecond As Double) As Integer
B_7443_sd_stop (ByVal AxisNo As Integer, ByVal Tdec As Double) As
                  Integer
B_7443_emg_stop (ByVal AxisNo As Integer) As Integer
B_7443_fix_speed_range (ByVal AxisNo As Integer, ByVal MaxVel As
                  Double) As Integer
B_7443_unfix_speed_range (ByVal AxisNo As Integer) As Integer
B_7443_get_current_speed (ByVal AxisNo As Integer, Speed As Double) As
                  Integer
B_7443_verify_speed  (ByVal StrVel As Double, ByVal MaxVel As Double,
                  minAccT As Double, maxAccT As Double, ByVal
                  MaxSpeed As Double) As Double
```

@パラメータ

AxisNo: 軸番号
StrVel: 移動開始速度(pps)
MaxVel: 移動最高速度(pps)
Tacc: 加速時間(sec)
SVacc: Ｓ字曲線加速時間のＳ字部分の速度(pps)
注意: SVacc = 0 : Ｓ字曲線だけの加速
NewVel: 変更後（新しい）の最高速度(pps)
Tdec: 減速時間(sec)
***Speed:** 現在の速度を保存するにおいての変数
(速度範囲: 0 ～ 6553500)
miniAccT: 加速時間の最小設定時間(sec)
maxAccT: 加速時間の最大設定時間(sec)
MaxSpeed: 最高速度(pps)

@復帰コード

ERR_NoError	: 0
ERR_SpeedError	: 11
ERR_SpeedChangeError	: 29
ERR_SlowDownPointError	: 16
ERR_AxisAlreadyStop	: 14

6.6　１軸のﾌﾟﾘｾｯﾄﾓｰﾄﾞ動作

@名前

_7443_start_tr_move	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでの台形動作の実行
_7443_start_ta_move	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでの台形動作の実行
_7443_start_sr_move	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでのＳ字曲線動作の実行
_7443_start_sa_move	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでのＳ字曲線動作の実行
_7443_set_move_ratio	ｺﾏﾝﾄﾞﾊﾟﾙｽとﾌｨｰﾄﾞﾊﾞｯｸﾊﾟﾙｽの比率設定
_7443_p_change	位置のｵｰﾊﾞｰﾗｲﾄﾞ機能
_7443_set_pcs_logic	PCS信号の論理設定
_7443_set_sd_pin	SD/PCS信号の設定
_7443_backlash_comp	ﾊﾞｯｸﾗｯｼｭ補正ﾊﾟﾙｽの設定
_7443_suppress_vibration	振動抑制時間の設定
_7443_set_idle_pulse	ｱｲﾄﾞﾘﾝｸﾞﾊﾟﾙｽの設定
_7443_set_auto_rdp	ﾀﾞｳﾝﾎﾟｲﾝﾄの手動/自動設定

@記述

全般

軸の動作方向は、ﾊﾟﾗﾒｰﾀの ***Pos*** か ***Dist*** の符号によって決められます。もし、移動距離が指定された速度に達することができないくらい短ければ、ｺﾝﾄﾛｰﾗは自動的に ***MaxVel***(最高速度)を下げます。更に、***Tacc***、***Tdec***、***SVacc***、***SVdec*** も dV/dt(加速/減速)と d(dV/dt)/dt(加速度の変化)が変化しない状態を維持するように短くなります。

_7443_start_tr_move

指定されたｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞによる距離を移動させるのに、台形動作で開始速度から最高速度まで加速し、減速して停止させます。加速時間と減速時間はそれぞれ指定できます。このﾌｧﾝｸｼｮﾝは軸の完了を待たずにﾌﾟﾛｸﾞﾗﾑの次のｽﾃｯﾌﾟを実行します。

_7443_start_ta_move

指定されたｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞによる位置に移動させるのに、台形動作で開始速度から最高速度まで加速し、減速して停止させます。加速時間と減速時間はそれぞれ指定できます。このﾌｧﾝｸｼｮﾝは軸の完了を待たずにﾌﾟﾛｸﾞﾗﾑの次のｽﾃｯﾌﾟを実行します。

_7443_start_sr_move

指定されたｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞによる距離を移動させるのに、Ｓ字曲線動作で開始速度から最高速度まで加速し、減速して停止させます。加速時間と減速時間はそれぞれ指定できます。このﾌｧﾝｸｼｮﾝは軸の完了を待たずにﾌﾟﾛｸﾞﾗﾑの次のｽﾃｯﾌﾟを実行します。

_7443_start_sa_move

は指定されたｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞによる位置に移動させるのに、Ｓ字曲線動作で開始速度から最高速度まで加速し、減速して停止させます。加速時間と減速時間はそれぞれ指定できます。このﾌｧﾝｸｼｮﾝは軸の完了を待たずにﾌﾟﾛｸﾞﾗﾑの次のｽﾃｯﾌﾟを実行します。

_7443_set_move_ratio

指定された軸の倍率を設定します。通常、機械的な分解能が異なっている場合、軸はこの倍率だけを設定します。例えば、ﾌｨｰﾄﾞﾊﾞｯｸｾﾝｻの分解能がｺﾏﾝﾄﾞﾊﾟﾙｽの２倍の分解能の場合、「ratio = 2」と設定します。

_7443_p_change

軸の動作中に目標位置を変えるのに使用します。このﾌｧﾝｸｼｮﾝにはいくつかの制限があります。使用の前にｾｸｼｮﾝ 4.6.2 をご参照下さい。

_7443_set_pcs_logic

PCS(Position Change Signal)信号の論理を設定するのに使用します。PCS 信号は SD 信号と同じﾋﾟﾝを共有します。SD/PCS ﾋﾟﾝが ***_7443_set_sd_pin()***ｺﾏﾝﾄﾞによって PCS に設定されたときだけ有効になります。

_7443_set_sd_pin

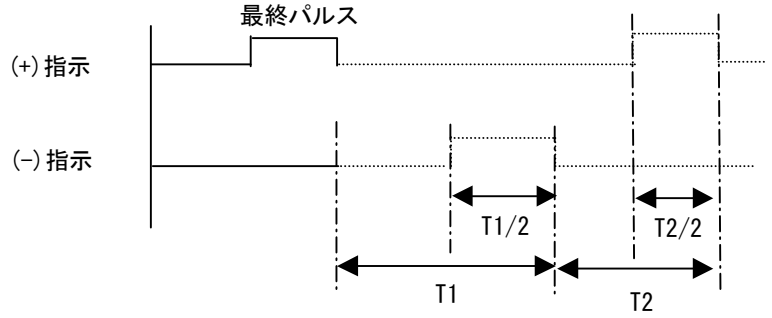
SD ﾋﾟﾝの操作ﾓｰﾄﾞを設定するのに使用します。SD/PCS ﾋﾟﾝはｽﾛｰﾀﾞｳﾝ信号(SD)か位置変更信号(PCS)のどちらかで使用します。ｾｸｼｮﾝ 4.3.1 をご参照下さい。

_7443_backlash_comp

ﾊﾞｯｸﾗｯｼｭ補正ﾊﾟﾙｽを設定します。方向指示の変更が起こる度に、PPCI7443 はｺﾏﾝﾄﾞを実行する前に、ﾊﾞｯｸﾗｯｼｭ補正ﾊﾟﾙｽを出力します。

_7443_suppress_vibration

このﾌｧﾝｸｼｮﾝはｺﾏﾝﾄﾞ動作完了後の機械の振動を抑制するのに使用します。ｺﾏﾝﾄﾞ動作完了直後にｺﾏﾝﾄﾞ信号と逆のﾊﾟﾙｽを１ﾊﾟﾙｽ出力し、そして、ｺﾏﾝﾄﾞﾊﾟﾙｽと同方向に１ﾊﾟﾙｽ出力します。

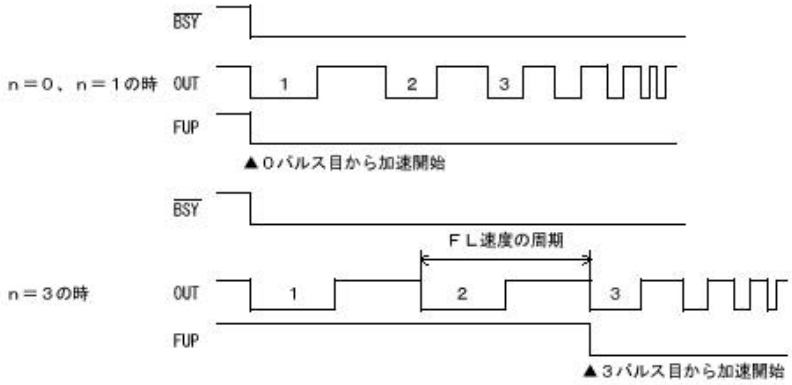


_7443_set_idle_pulse

機械の振動を抑制するためのｱｲﾄﾞﾘﾝｸﾞﾊﾟﾙｽを設定します。移動開始時に開始速度(StrVel)で数ﾊﾟﾙｽｱｲﾄﾞﾘﾝｸﾞ出力してからか加速を開始させます。

注意：このﾌｧﾝｸｼｮﾝを使用する場合は、設定値に 2～7 を設定して下さい。

使用しない場合には 0 か 1 を設定して下さい。



_7443_set_auto_rdp

軸動作におけるﾀﾞｳﾝﾎﾟｲﾝﾄ設定を手動で行うか自動で行うかの設定をします。手動、自動どちらに設定しても位置決め動作のｺﾏﾝﾄﾞに影響はありません。ｺﾏﾝﾄﾞﾗｲﾌﾞﾗﾘの内部計算が異なるだけです。

ただし、自動に設定した場合、加速と減速に設定する値を

（減速時間）≦（加速時間×２）の範囲で設定して下さい。

@構文

C/C++(DOS、Windows95/NT)

```
I16 _7443_start_tr_move(I16 AxisNo, F64 Dist, F64 StrVel, F64 MaxVel, F64
                        Tacc,F64 Tdec);
I16 _7443_start_ta_move(I16 AxisNo, F64 Pos, F64 StrVel, F64 MaxVel, F64
                        Tacc, F64 Tdec);
I16 _7443_start_sr_move(I16 AxisNo, F64 Dist, F64 StrVel, F64 MaxVel, F64
                        Tacc, F64 Tdec, F64 SVacc, F64 SVdec);
I16 _7443_start_sa_move(I16 AxisNo, F64 Pos, F64 StrVel, F64 MaxVel,
                        F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);
I16 _7443_set_move_ratio(I16 AxisNo, F64 move_ratio);
I16 _7443_p_change(I16 AxisNo, F64 NewPos);
I16 _7443_set_pcs_logic(I16 AxisNo, I16 pcs_logic);
I16 _7443_set_sd_pin(I16 AxisNo, I16 Type);
```

I16 _7443_backlash_comp(I16 AxisNo, I16 BCompPulse);
I16 _7443_suppress_vibration(I16 AxisNo, U16 T1, U16 T2);
I16 _7443_set_idle_pulse(I16 AxisNo,I16 idl_pulse)
I16 _7443_set_auto_rdp(I16 CardNo,I16 on_off)

Visual Basic(Windows95/NT)

B_7443_start_tr_move (ByVal AxisNo As Integer, ByVal Dist As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_7443_start_ta_move (ByVal AxisNo As Integer, ByVal Pos As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_7443_start_sr_move (ByVal AxisNo As Integer, ByVal Dist As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_7443_start_sa_move (ByVal AxisNo As Integer, ByVal Pos As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_7443_set_move_ratio (ByVal AxisNo As Integer, ByVal move_ratio As Double) As Integer

B_7443_p_change (ByVal AxisNo As Integer, ByVal NewPos As Double) As Integer

B_7443_set_pcs_logic (ByVal AxisNo As Integer, ByVal pcs_logic As Integer) As Integer

B_7443_set_sd_pin (ByVal AxisNo As Integer, ByVal Type As Integer) As Integer

B_7443_backlash_comp (ByVal AxisNo As Integer, ByVal BCompPulse As Integer, ByVal ForwardTime As Integer) As Integer

B_7443_suppress_vibration (ByVal AxisNo As Integer, ByVal ReserveTime As Integer, ByVal ForwardTime As Integer) As Integer

B_7443_set_idle_pulse (ByVal AxisNo As Integer,ByVal idl_pulse As Integer) As Integer

B_7443_set_auto_rdp (ByVal CardNo As Integer,ByVal on_off As Integer) As Integer

@パラメータ

AxisNo:	軸番号
Dist:	移動距離
Pos:	ｱﾌﾞｿﾘｭｰﾄによる位置
StrVel:	移動開始速度(pps)
MaxVel:	移動最高速度(pps)
Tacc:	加速時間(sec)
Tdec:	減速時間(sec)
SVacc:	Ｓ字曲線加速時間のＳ字部分の速度(pps) 注意:Ｓ字曲線のみの加速の場合 SVacc = 0 とする
SVdec:	Ｓ字曲線減速時間のＳ字部分の速度(pps) 注意:Ｓ字曲線のみの減速の場合 SVdec = 0 とする
Move_ratio:	ﾌｨｰﾄﾞﾊﾞｯｸ分解能/ｺﾏﾝﾄﾞ分解能の比率、「0」以外を設定
NewPos:	位置のｵｰﾊﾞｰﾗｲﾄﾞのための新しいｱﾌﾞｿﾘｭｰﾄ位置
BcompPulse:	ﾊﾞｯｸﾗｯｼｭ補正ﾊﾟﾙｽ
ReserveTime,T1:	振動抑制機能の逆ﾊﾟﾙｽ出力時間
ForwardTime,T2:	振動抑制機能の正ﾊﾟﾙｽ出力時間
pcs_logic:	pcs 信号の論理設定 pcs_logic=0：Active Low pcs_logic=1：Active High
Type:	sd/pcs ﾋﾟﾝの選択 Type=0：SD Type=1：PCS
idl_pulse:	ｱｲﾄﾞﾘﾝｸﾞﾊﾟﾙｽ (設定範囲： 0 ～ 7)
on_off:	0：手動設定（初期値） 1：自動設定

@復帰コード

ERR_NoError	： 0
ERR_SpeedError	: 11
ERR_PchangeSlowDownPointError	: 28
ERR_MoveRatioError	: 12

6.7　直線補間動作

@名前

_7443_start_tr_move_xy	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでの台形動作によるXとYの２軸直線補間の実行
_7443_start_ta_move_xy	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでの台形動作によるXとYの２軸直線補間の実行
_7443_start_sr_move_xy	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでのＳ字曲線動作によるXとYの２軸直線補間の実行
_7443_start_sa_move_xy	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでのＳ字曲線動作によるXとYの２軸直線補間の実行
_7443_start_tr_move_zu	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでの台形動作によるZとUの２軸直線補間の実行
_7443_start_ta_move_zu	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでの台形動作によるZとUの２軸直線補間の実行
_7443_start_sr_move_zu	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでのＳ字曲線動作によるZとUの２軸直線補間の実行
_7443_start_sa_move_zu	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでのＳ字曲線動作によるZとUの２軸直線補間の実行
_7443_start_tr_line2	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでの台形動作による任意の２軸直線補間の実行
_7443_start_sr_line2	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでのＳ字曲線動作による任意の２軸直線補間の実行
_7443_start_ta_line2	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでの台形動作による任意の２軸直線補間の実行
_7443_start_sa_line2	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでのＳ字曲線動作による任意の２軸直線補間の実行
_7443_start_tr_line3	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでの台形動作による任意の３軸直線補間の実行
_7443_start_sr_line3	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでのＳ字曲線動作による任意の３軸直線補間の実行
_7443_start_ta_line3	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでの台形動作による任意の３軸直線補間の実行
_7443_start_sa_line3	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでのＳ字曲線動作による任意の３軸直線補間の実行
_7443_start_tr_line4	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでの台形動作による４軸直線補間の実行
_7443_start_sr_line4	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでのＳ字曲線動作による４軸直線補間の実行

_7443_start_ta_line4 ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでの台形動作による４軸直線補間の実行

_7443_start_sa_line4 ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでのＳ字曲線動作による４軸直線補間の実行

_7443_set_line_move_mode 連続直線補間の設定

_7443_set_axis_option 補間動作の速度モードの選択

@記述

ﾌｧﾝｸｼｮﾝ	軸数	加減速動作	ｲﾝｸﾘﾒﾝﾀﾙ/ｱﾌﾞｿﾘｭｰﾄ	指示軸
_7443_start_tr_move_xy	2	T	R	軸０，１
_7443_start_ta_move_xy	2	T	A	軸０，１
_7443_start_sr_move_xy	2	S	R	軸０，１
_7443_start_sa_move_xy	2	S	A	軸０，１
_7443_start_tr_move_zu	2	T	R	軸２，３
_7443_start_ta_move_zu	2	T	A	軸２，３
_7443_start_sr_move_zu	2	S	R	軸２，３
_7443_start_sa_move_zu	2	S	A	軸２，３
_7443_start_tr_line2	2	T	R	任意の２軸
_7443_start_ta_line2	2	T	A	任意の２軸
_7443_start_sr_line2	2	S	R	任意の２軸
_7443_start_sa_line2	2	S	A	任意の２軸
_7443_start_tr_line3	3	T	R	任意の３軸
_7443_start_ta_line3	3	T	A	任意の３軸
_7443_start_sr_line3	3	S	R	任意の３軸
_7443_start_sa_line3	3	S	A	任意の３軸
_7443_start_tr_line4	4	T	R	４軸
_7443_start_ta_line4	4	T	A	４軸
_7443_start_sr_line4	4	S	R	４軸
_7443_start_sa_line4	4	S	A	４軸

T：台形動作　S：Ｓ字曲線動作　R：ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞ　A：ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞ

_7443_set_line_move_mode()

直線補間動作を連続させるﾓｰﾄﾞの設定をします。このﾓｰﾄﾞが
連続に設定されている場合、停止ｺﾏﾝﾄﾞが書き込まれるまで停止しません。

@構文

C/C++(DOS、Windows95/NT)

```
I16 _7443_start_tr_move_xy(I16 CardNo, F64 DistX, F64 DistY, F64 StrVel,
                           F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _7443_start_ta_move_xy(I16 CardNo, F64 PosX, F64 PosY, F64 StrVel,
                           F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _7443_start_sr_move_xy(I16 CardNo, F64 DistX, F64 DistY, F64 StrVel,
                           F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64
                           SVdec);
I16 _7443_start_sa_move_xy(I16 CardNo, F64 PosX, F64 PosY, F64 StrVel,
                           F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64
                           SVdec);
I16 _7443_start_tr_move_zu(I16 CardNo, F64 DistX, F64 DistY, F64 StrVel,
                           F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _7443_start_ta_move_zu(I16 CardNo, F64 PosX, F64 PosY, F64 StrVel,
                           F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _7443_start_sr_move_zu(I16 CardNo, F64 DistX, F64 DistY, F64 StrVel,
                           F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64
                           SVdec);
I16 _7443_start_sa_move_zu(I16 CardNo, F64 PosX, F64 PosY, F64 StrVel,
                           F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64
                           SVdec);
I16 _7443_start_tr_line2(I16 CardNo, I16 *AxisArray, F64 DistX, F64 DistY,
                           F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _7443_start_ta_line2(I16 CardNo, I16 *AxisArray, F64 PosX, F64 PosY,
                           F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _7443_start_sr_line2(I16 CardNo, I16 *AxisArray, F64 DistX, F64 DistY,
                           F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64
                           SVacc, F64 SVdec);
I16 _7443_start_sa_line2(I16 CardNo, I16 *AxisArray, F64 PosX, F64 PosY,
                           F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64
                           SVacc, F64 SVdec);
I16 _7443_start_tr_line3(I16 CardNo, I16 *AxisArray, F64 DistX, F64 DistY,
                           F64 DistZ, F64 StrVel, F64 MaxVel, F64 Tacc, F64
                           Tdec);
I16 _7443_start_ta_line3(I16 CardNo, I16 *AxisArray, F64 PosX, F64 PosY,
                           F64 PosZ, F64 StrVel, F64 MaxVel, F64 Tacc, F64
                           Tdec);
I16 _7443_start_sr_line3(I16 CardNo, I16 *AxisArray, F64 DistX, F64 DistY,
                           F64 DistZ, F64 StrVel, F64 MaxVel, F64 Tacc, F64
                           Tdec, F64 SVacc, F64 SVdec);
I16 _7443_start_sa_line3(I16 CardNo, I16 *AxisArray, F64 PosX, F64 PosY,
                           F64 PosZ, F64 StrVel, F64 MaxVel, F64 Tacc, F64
                           Tdec, F64 SVacc, F64 SVdec);
I16 _7443_start_tr_line4(I16 CardNo, F64 DistX, F64 DistY, F64 DistZ, F64
                           DistU, F64 StrVel, F64 MaxVel, F64 Tacc, F64
                           Tdec);
I16 _7443_start_ta_line4(I16 CardNo, F64 PosX, F64 PosY, F64 PosZ, F64
                           PosU, F64 StrVel, F64 MaxVel, F64 Tacc, F64
                           Tdec);
```

I16 _7443_start_sr_line4(I16 CardNo, F64 DistX, F64 DistY, F64 DistZ, F64 DistU, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);
I16 _7443_start_sa_line4(I16 CardNo, F64 PosX, F64 PosY, F64 PosZ, F64 PosU, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);
I16 _7443_set_line_move_mode(I16 AxisNo, I16 Mode);
I16 _7443_set_axis_option(I16 AxisNo, I16 option);

Visual Basic(Windows95/NT)

B_7443_start_tr_move_xy (ByVal CardNo As Integer, ByVal Dist As Double, ByVal Dist As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer
B_7443_start_ta_move_xy (ByVal CardNo As Integer, ByVal Pos As Double, ByVal Pos As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer
B_7443_start_sr_move_xy (ByVal CardNo As Integer, ByVal Dist As Double, ByVal Dist As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer
B_7443_start_sa_move_xy (ByVal CardNo As Integer, ByVal Pos As Double, ByVal Pos As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer
B_7443_start_tr_move_zu (ByVal CardNo As Integer, ByVal Dist As Double, ByVal Dist As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer
B_7443_start_ta_move_zu (ByVal CardNo As Integer, ByVal Pos As Double, ByVal Pos As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer
B_7443_start_sr_move_zu (ByVal CardNo As Integer, ByVal Dist As Double, ByVal Dist As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer
B_7443_start_sa_move_zu (ByVal CardNo As Integer, ByVal Pos As Double, ByVal Pos As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer
B_7443_start_tr_line2 (ByVal CardNo As Integer, AxisArray As Integer, ByVal DistX As Double, ByVal DistY As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_7443_start_ta_line2 (ByVal CardNo As Integer, AxisArray As Integer,
ByVal PosX As Double, ByVal PosY As Double,
ByVal StrVel As Double, ByVal MaxVel As Double,
ByVal Tacc As Double, ByVal Tdec As Double) As
Integer
B_7443_start_sr_line2 (ByVal CardNo As Integer, AxisArray As Integer,
ByVal DistX As Double, ByVal DistY As Double,
ByVal StrVel As Double, ByVal MaxVel As Double,
ByVal Tacc As Double, ByVal Tdec As Double,
ByVal SVacc As Double, ByVal SVdec As Double)
As Integer
B_7443_start_sa_line2 (ByVal CardNo As Integer, AxisArray As Integer,
ByVal PosX As Double, ByVal PosY As Double,
ByVal StrVel As Double, ByVal MaxVel As Double,
ByVal Tacc As Double, ByVal Tdec As Double,
ByVal SVacc As Double, ByVal SVdec As Double)
As Integer
B_7443_start_tr_line3 (ByVal CardNo As Integer, AxisArray As Integer,
ByVal DistX As Double, ByVal DistY As Double,
ByVal DistZ As Double, ByVal StrVel As Double,
ByVal MaxVel As Double, ByVal Tacc As Double,
ByVal Tdec As Double) As Integer
B_7443_start_ta_line3 (ByVal CardNo As Integer, AxisArray As Integer,
ByVal PosX As Double, ByVal PosY As Double,
ByVal PosZ As Double, ByVal StrVel As Double,
ByVal MaxVel As Double, ByVal Tacc As Double,
ByVal Tdec As Double) As Integer
B_7443_start_sr_line3 (ByVal CardNo As Integer, AxisArray As Integer,
ByVal DistX As Double, ByVal DistY As Double,
ByVal DistZ As Double, ByVal StrVel As Double,
ByVal MaxVel As Double, ByVal Tacc As Double,
ByVal Tdec As Double, ByVal SVacc As Double,
ByVal SVdec As Double) As Integer
B_7443_start_sa_line3 (ByVal CardNo As Integer, AxisArray As Integer,
ByVal PosX As Double, ByVal PosY As Double,
ByVal PosZ As Double, ByVal StrVel As Double,
ByVal MaxVel As Double, ByVal Tacc As Double,
ByVal Tdec As Double, ByVal SVacc As Double,
ByVal SVdec As Double) As Integer
B_7443_start_tr_line4 (ByVal CardNo As Integer, ByVal DistX As Double,
ByVal DistY As Double, ByVal DistZ As Double,
ByVal DistU As Double, ByVal StrVel As Double,
ByVal MaxVel As Double, ByVal Tacc As Double,
ByVal Tdec As Double) As Integer
B_7443_start_ta_line4 (ByVal CardNo As Integer, ByVal PosX As Double,
ByVal PosY As Double, ByVal PosZ As Double,
ByVal PosU As Double, ByVal StrVel As Double,
ByVal MaxVel As Double, ByVal Tacc As Double,
ByVal Tdec As Double) As Integer

```
B_7443_start_sr_line4 (ByVal CardNo As Integer, ByVal DistX As Double,
                       ByVal DistY As Double, ByVal DistZ As Double,
                       ByVal DistU As Double, ByVal StrVel As Double,
                       ByVal MaxVel As Double, ByVal Tacc As Double,
                       ByVal Tdec As Double, ByVal SVacc As Double,
                       ByVal SVdec As Double) As Integer
B_7443_start_sa_line4 (ByVal CardNo As Integer, ByVal PosX As Double,
                       ByVal PosY As Double, ByVal PosZ As Double,
                       ByVal PosU As Double, ByVal StrVel As Double,
                       ByVal MaxVel As Double, ByVal Tacc As Double,
                       ByVal Tdec As Double, ByVal SVacc As Double,
                       ByVal SVdec As Double) As Integer
B_7443_set_line_move_mode (ByVal AxisNo As Integer, ByVal Mode As
                       Integer) As Integer
B_7443_set_axis_option (ByVal AxisNo As Integer, ByVal option1 As
                       Integer) As Integer
```

@パラメータ

CardNo: 基板番号
DistX: 軸０の移動距離
DistY: 軸１の移動距離
DistZ: 軸２の移動距離
DistU: 軸３の移動距離
PosX: 軸０のｱﾌﾞｿﾘｭｰﾄ位置
PosY: 軸１のｱﾌﾞｿﾘｭｰﾄ位置
PosZ: 軸２のｱﾌﾞｿﾘｭｰﾄ位置
PosU: 軸３のｱﾌﾞｿﾘｭｰﾄ位置
StrVel: 移動開始速度(pps)
MaxVel: 移動最高速度(pps)
Tacc: 加速時間(sec)
Tdec: 減速時間(sec)
SVacc: Ｓ字曲線加速時間のＳ字部分の速度(pps)
注意:Ｓ字曲線のみの加速の場合 SVacc = 0 とする
SVdec: Ｓ字曲線減速時間のＳ字部分の速度(pps)
注意:Ｓ字曲線のみの減速の場合 SVdec = 0 とする
AxisArray: 補間を実行する軸番号の配列
例: Int AxisArray2={0,2}; //軸 0 と 2
Int AxisArray3={0,1,3}; //軸 0 と 1 と 3
注意:もし n<m であるならば、AxisArray[n]は AxisArray[m]より
も小さくなければならない
Mode: 連続補間ﾓｰﾄﾞ
Mode=0： 位置決め補間ﾓｰﾄﾞ
Mode=1： 連続補間ﾓｰﾄﾞ
option1: 補間速度ﾓｰﾄﾞ
option1=0： 通常の補間速度ﾓｰﾄﾞ
option1=1： 合成速度一定制御ﾓｰﾄﾞ

@復帰コード

ERR_NoError	: 0
ERR_SpeedError	: 11
ERR_AxisArrayError	: 15

6.8　円弧補間動作

@名前

_7443_start_r_arc_xy	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでのXとYの円弧補間の実行
_7443_start_a_arc_xy	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでのXとYの円弧補間の実行
_7443_start_r_arc_zu	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでのZとUの円弧補間の実行
_7443_start_a_arc_zu	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでのZとUの円弧補間の実行
_7443_start_r_arc2	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでの任意の２軸の円弧補間の実行
_7443_start_a_arc2	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでの任意の２軸の円弧補間の実行
_7443_start_tr_arc_xy	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでの台形動作によるXとYの円弧補間の実行
_7443_start_ta_arc_xy	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでの台形動作によるXとYの円弧補間の実行
_7443_start_sr_arc_xy	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでのＳ字曲線動作によるXとYの円弧補間の実行
_7443_start_sa_arc_xy	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでのＳ字曲線動作によるXとYの円弧補間の実行
_7443_start_tr_arc_zu	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでの台形動作によるZとUの円弧補間の実行
_7443_start_ta_arc_zu	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでの台形動作によるZとUの円弧補間の実行
_7443_start_sr_arc_zu	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでのＳ字曲線動作によるZとUの円弧補間の実行
_7443_start_sa_arc_zu	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでのＳ字曲線動作によるZとUの円弧補間の実行
_7443_start_tr_arc2	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでの台形動作による任意の２軸円弧補間の実行
_7443_start_ta_arc2	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでの台形動作による任意の２軸円弧補間の実行
_7443_start_sr_arc2	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでのＳ字曲線動作による任意の２軸円弧補間の実行
_7443_start_sa_arc2	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞでのＳ字曲線動作による任意の２軸円弧補間の実行

@記述

ﾌｧﾝｸｼｮﾝ	加減速動作	ｲﾝｸﾘﾒﾝﾀﾙ/ｱﾌﾞｿﾘｭｰﾄ	指示軸	ﾊｰﾄﾞｳｪｱﾊﾞｰｼﾞｮﾝ(4 桁目)
_7443_start_r_arc_xy	なし	R	軸０と１	Ｏor １
_7443_start_a_arc_xy	なし	A	軸０と１	Ｏor １
_7443_start_r_arc_zu	なし	R	軸２と３	Ｏor １
_7443_start_a_arc_zu	なし	A	軸２と３	Ｏor １
_7443_start_r_arc2	なし	R	任意の２軸	Ｏor １
_7443_start_a_arc2	なし	A	任意の２軸	Ｏor １
_7443_start_tr_arc_xy	T	R	軸０と１	1
_7443_start_ta_arc_xy	T	A	軸０と１	1
_7443_start_sr_arc_xy	S	R	軸０と１	1
_7443_start_sa_arc_xy	S	A	軸０と１	1
_7443_start_tr_arc_zu	T	R	軸２と３	1
_7443_start_ta_arc_zu	T	A	軸２と３	1
_7443_start_sr_arc_zu	S	R	軸２と３	1
_7443_start_sa_arc_zu	S	A	軸２と３	1
_7443_start_tr_arc2	T	R	任意の２軸	1
_7443_start_ta_arc2	T	A	任意の２軸	1
_7443_start_sr_arc2	S	R	任意の２軸	1
_7443_start_sa_arc2	S	A	任意の２軸	1

T：台形動作　S：S字曲線動作　R：ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞ　A：ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞ

@構文

C/C++(DOS、Windows95/NT)

```
I16 _7443_start_r_arc_xy(I16 CardNo, F64 OffsetCx, F64 OffsetCy, F64
                         OffsetEx, F64 OffsetEy, I16 DIR, F64 MaxVel);
I16 _7443_start_a_arc_xy(I16 CardNo, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16
                         DIR, F64 MaxVel);
I16 _7443_start_r_arc_zu(I16 CardNo, F64 OffsetCx, F64 OffsetCy, F64
                         OffsetEx, F64 OffsetEy, I16 DIR, F64 MaxVel);
I16 _7443_start_a_arc_zu(I16 CardNo, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16
                         DIR, F64 MaxVel);
I16 _7443_start_r_arc2(I16 CardNo, I16 *AxisArray, F64 OffsetCx, F64
                         OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64
                         MaxVel);
I16 _7443_start_a_arc2(I16 CardNo, I16 *AxisArray, F64 Cx, F64 Cy, F64 Ex,
                         F64 Ey, I16 DIR, F64 MaxVel);

I16 _7443_start_tr_arc_xy(I16 CardNo, F64 OffsetCx, F64 OffsetCy, F64
                         OffsetEx, F64 OffsetEy, I16 DIR,  F64 StrVel, F64
                         MaxVel, F64 Tacc,    F64 Tdec);
```

I16 _7443_start_ta_arc_xy(I16 CardNo, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 DIR, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);

I16 _7443_start_sr_arc_xy(I16 CardNo, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);

I16 _7443_start_sa_arc_xy(I16 CardNo, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 DIR, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);

I16 _7443_start_tr_arc_zu(I16 CardNo, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);

I16 _7443_start_ta_arc_zu(I16 CardNo, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 DIR, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);

I16 _7443_start_sr_arc_zu(I16 CardNo, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);

I16 _7443_start_sa_arc_zu(I16 CardNo, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 DIR, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);

I16 _7443_start_tr_arc2(I16 CardNo, I16 *AxisArray, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);

I16 _7443_start_ta_arc2(I16 CardNo, I16 *AxisArray, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 DIR, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);

I16 _7443_start_sr_arc2(I16 CardNo, I16 *AxisArray, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);

I16 _7443_start_sa_arc2(I16 CardNo, I16 *AxisArray, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 DIR, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);

Visual Basic(Windows95/NT)

B_7443_start_a_arc_xy (ByVal CardNo As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer, ByVal MaxVel As Double) As Integer

B_7443_start_r_arc_xy (ByVal CardNo As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal MaxVel As Double) As Integer

B_7443_start_a_arc_zu (ByVal CardNo As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer, ByVal MaxVel As Double) As Integer

B_7443_start_r_arc_zu (ByVal CardNo As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal MaxVel As Double) As Integer

B_7443_start_a_arc2 (ByVal CardNo As Integer, AxisArray As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer, ByVal MaxVel As Double) As Integer

B_7443_start_r_arc2 (ByVal CardNo As Integer, AxisArray As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal MaxVel As Double) As Integer

B_7443_start_tr_arc_xy(ByVal CardNo As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double);

B_7443_start_ta_arc_xy(ByVal CardNo As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double);

B_7443_start_sr_arc_xy(ByVal CardNo As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double);

B_7443_start_sa_arc_xy(ByVal CardNo As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double);

B_7443_start_tr_arc_zu(ByVal CardNo As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double);

B_7443_start_ta_arc_zu(ByVal CardNo As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double);

B_7443_start_sr_arc_zu(ByVal CardNo As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double);

B_7443_start_sa_arc_zu(ByVal CardNo As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double);

B_7443_start_tr_arc2 (ByVal CardNo As Integer, AxisArray As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_7443_start_ta_arc2 (ByVal CardNo As Integer, AxisArray As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer,ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_7443_start_sr_arc2 (ByVal CardNo As Integer, AxisArray As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_7443_start_sa_arc2 (ByVal CardNo As Integer, AxisArray As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer,ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

@パラメータ

CardNo: 基板番号
OffsetCx: 中心のＸ軸ｵﾌｾｯﾄ量
OffsetCy: 中心のＹ軸ｵﾌｾｯﾄ量
OffsetEx: 円弧終点のＸ軸ｵﾌｾｯﾄ量
OffsetEy: 円弧終点のＹ軸ｵﾌｾｯﾄ量
Cx: 中心のＸ軸のｱﾌﾞｿﾘｭｰﾄ位置
Cy: 中心のＹ軸のｱﾌﾞｿﾘｭｰﾄ位置
Ex: 円弧終点のＸ軸のｱﾌﾞｿﾘｭｰﾄ位置
Ey: 円弧終点のＹ軸のｱﾌﾞｿﾘｭｰﾄ位置
DIR: 円弧の方向
DIR = 0：CW, DIR = 1：CCW
StrVel: 移動開始速度(pps)
MaxVel: 移動最高速度(pps)
Tacc: 加速時間(sec)
Tdec: 減速時間(sec)
SVacc: Ｓ字曲線加速時間のＳ字部分の速度(pps)
注意:Ｓ字曲線のみの加速の場合 SVacc = 0 とする
SVdec: Ｓ字曲線減速時間のＳ字部分の速度(pps)
注意:Ｓ字曲線のみの減速の場合 SVdec = 0 とする
AxisArray: 補間動作を実行する軸番号の配列
例: Int AxisArray2={0,2}; //軸 0 と 2
Int AxisArray2={1,3}; //軸 1 と 3
注意:AxisArray[0]は AxisArray[1]よりも小さくなければならない。

@復帰コード

ERR_NoError : 0
ERR_SpeedError : 11
ERR_AxisArrayError : 15

6.9 原点復帰

@名前

_7443_set_home_config	原点復帰ﾓｰﾄﾞ、論理等の設定
_7443_home_move	原点復帰動作の実行
_7443_escape_home	原点抜け出し動作の実行
_7443_home_search	原点ｻｰﾁ動作の実行(ｵﾌｾｯﾄ機能なし)
_7443_auto_home_search	原点ｻｰﾁ動作の実行(ｵﾌｾｯﾄ機能付き)

@記述

_7443_set_home_config

原点復帰動作のための、原点復帰ﾓｰﾄﾞ、原点信号(ORG)およびＺ相信号(EZ)の論理、Ｚ相のｶｳﾝﾄ数そして、ERC 信号の出力選択を構成します。原点復帰ﾓｰﾄﾞの設定はｾｸｼｮﾝ 4.1.8 をご参照下さい。

_7443_home_move

_7443_set_home_config()ｺﾏﾝﾄﾞの設定に従って、このﾌｧﾝｸｼｮﾝは軸の原点復帰動作を実行します。動作方向は速度ﾊﾟﾗﾒｰﾀ ***StrVel,MaxVel*** の符号によります。このﾌｧﾝｸｼｮﾝの停止条件が ***home_mode*** によって決まるので、ﾕｰｻﾞｰは、最初の移動方向の選択には注意しなければなりません。また、ﾘﾐｯﾄｽｲｯﾁが ON しているかあるいは他の条件で軸が停止しているとき、ﾕｰｻﾞｰはその状態の処理に十分注意しなければなりません。

_7443_home_move()ｺﾏﾝﾄﾞで原点復帰動作中に***_7443_v_stop()***ｺﾏﾝﾄﾞを実行すると、軸を停止させることができます。

_7443_escape_home

原点信号が ON しているときに原点信号が OFF する位置まで移動させます。原点信号が OFF しているときには何も動作しません。

原点信号が ON しているときに***_7443_escape_home()***ｺﾏﾝﾄﾞを実行すると指定方向に移動し、原点信号が OFF したら 1ﾊﾟﾙｽ出力して減速停止します。

_7443_home_serch,_7443_auto_home_search

原点復帰動作の拡張動作です。原点信号が ON しているときの原点復帰動作や原点復帰動作が EL 信号が ON して正常に停止できなかった場合に使用します。

原点信号が OFF の時に実行した場合は、***_7443_home_move()***ｺﾏﾝﾄﾞと同じになります。また、原点信号が ON する前に EL 信号が ON した場合は、自動的に逆方向へ移動開始し、原点信号が ON して OFF するまで移動し、それから***_7443_home_move()***ｺﾏﾝﾄﾞと同じ動作となります。原点信号が ON しているときに実行すると、原点信号が OFF する位置まで移動開始し、それから***_7443_home_move()***ｺﾏﾝﾄﾞと同じ動作となります。

_7443_home_search()と_7443_auto_home_search()ｺﾏﾝﾄﾞの違いは原点

信号が ON しているときに OFF させるｵﾌｾｯﾄ移動量が設定できるかできないかの違いです。_7443_home_search()ｺﾏﾝﾄﾞに OFF させる移動量の設定がありません。直前に動作させた移動量がそのまま OFF させる移動量となります。

@構文

C/C++(DOS、Windows95/NT)

```
I16 _7443_set_home_config(I16 AxisNo, I16 home_mode, I16 org_logic, I16
                ez_logic, I16 ez_count, I16 erc_out);
I16 _7443_home_move(I16 AxisNo, F64 StrVel, F64 MaxVel, F64 Tacc);
I16 _7443_escape_home(I16 AxisNo, F64 StrVel, F64 MaxVel, F64 Tacc);
I16 _7443_home_search(I16 AxisNo, F64 StrVel, F64 MaxVel, F64 Tacc);
I16 _7443_auto_home_search(I16 AxisNo, F64 StrVel, F64 MaxVel, F64
                Tacc, F64 ORGOffset);
```

Visual Basic(Windows95/NT)

```
B_7443_set_home_config (ByVal AxisNo As Integer, ByVal home_mode As
                Integer, ByVal org_logic As Integer, ByVal ez_logic
                As Integer, ByVal ez_count As Integer, ByVal
                erc_out As Integer) As Integer
B_7443_home_move (ByVal AxisNo As Integer, ByVal StrVel As Double,
                ByVal MaxVel As Double, ByVal Tacc As Double)
                As Integer
B_7443_escape_home (ByVal AxisNo As Integer, ByVal StrVel As Double,
                ByVal MaxVel As Double, ByVal Tacc As Double)
                As Integer
B_7443_home_search (ByVal AxisNo As Integer, ByVal StrVel As Double,
                ByVal MaxVel As Double, ByVal Tacc As Double)
                As Integer
B_7443_auto_home_search (ByVal AxisNo As Integer, ByVal StrVel As
                Double, ByVal MaxVel As Double, ByVal Tacc As
                Double, Byval ORGOffset As Double) As Integer
```

@パラメータ

AxisNo: 軸番号
home_mode: 原点復帰ﾓｰﾄﾞ(0 ～ 12)
(セクション 4.1.8 をご参照下さい)
org_logic: ORG 信号の論理設定
org_logic=0：Active Low
org_logic=1：Active High
EZ_logic: EZ 信号の論理設定
EZ_logic=0： Active Low
EZ_logic=1： Active High
ez_count: 0 ～ 15 (セクション 4.1.8 をご参照下さい)
erc_out: ERC 出力選択
erc_out=0 出力なし
erc_out=1 原点復帰完了で出力
StrVel: 移動開始速度(pps)
MaxVel: 移動最高速度(pps)
Tacc: 加速時間(sec)
ORGOffset: 原点信号が ON していた場合に OFF させる為の移動量

@復帰コード

ERR_NoError : 0

6.10　パルサー動作

@名前

_7443_set_pulser_iptmode	ﾊﾟﾙｻｰ入力ﾓｰﾄﾞの設定
_7443_pulser_vmove	ﾊﾟﾙｻｰ速度移動の実行
_7443_pulser_pmove	ﾊﾟﾙｻｰ位置移動の実行
_7443_pulser_home_move	ﾊﾟﾙｻｰ原点復帰動作の実行
_7443_pulser_r_line2	ﾊﾟﾙｻｰによる２軸直線補間
_7443_pulser_r_arc2	ﾊﾟﾙｻｰによる２軸円弧補間

@記述

_7443_set_pulser_iptmode

手動ﾊﾟﾙｻｰの入力ﾓｰﾄﾞを設定するのに使用します。

_7443_pulser_vmove

このｺﾏﾝﾄﾞが書かれると手動のﾊﾟﾙｻｰ入力に従って軸を動かし始めます。軸はﾊﾟﾙｻｰから１ﾊﾟﾙｽ受け取ると１ﾊﾟﾙｽ出力します。***sd_stop***ｺﾏﾝﾄﾞか***emg_stop***ｺﾏﾝﾄﾞが書かれるまでﾓｰﾄﾞは継続します。

_7443_pulser_pmove

このｺﾏﾝﾄﾞが書かれると手動のﾊﾟﾙｻｰ入力に従って軸を動かし始めます。軸はﾊﾟﾙｻｰから１ﾊﾟﾙｽ受け取ると１ﾊﾟﾙｽ出力します。***sd_stop***ｺﾏﾝﾄﾞか***emg_stop***ｺﾏﾝﾄﾞが書かれるか、あるいは出力ﾊﾟﾙｽ数がdistに達するまでﾓｰﾄﾞを継続します。

_7443_pulser_home_move

このｺﾏﾝﾄﾞが書かれると手動のﾊﾟﾙｻｰ入力に従って軸を動かし始めます。軸はﾊﾟﾙｻｰから１ﾊﾟﾙｽ受け取ると１ﾊﾟﾙｽ出力します。***sd_stop***ｺﾏﾝﾄﾞか***emg_stop***ｺﾏﾝﾄﾞが書かれるか、あるいは原点移動完了するまでﾓｰﾄﾞを継続します。

_7443_pulser_r_line2

手動ﾊﾟﾙｻｰによる２軸直線補間を実行します。詳細はｾｸｼｮﾝ4.1.13をご参照下さい。***sd_stop***ｺﾏﾝﾄﾞか***emg_stop***ｺﾏﾝﾄﾞが書かれるか、あるいは出力ﾊﾟﾙｽ数がdistに達するまでﾓｰﾄﾞを継続します。

_7443_pulser_r_arc2

手動ﾊﾟﾙｻｰによる２軸円弧補間を実行します。詳細はｾｸｼｮﾝ4.1.13をご参照下さい。***sd_stop***ｺﾏﾝﾄﾞか***emg_stop***ｺﾏﾝﾄﾞが書かれるか、あるいは出力ﾊﾟﾙｽ数がdistに達するまでﾓｰﾄﾞを継続します。

@構文

C/C++(DOS、Windows95/NT)

```
I16 _7443_set_pulser_iptmode(I16 AxisNo,I16 InputMode, I16 Inverse);
I16 _7443_pulser_vmove(I16 AxisNo, F64 SpeedLimit);
I16 _7443_pulser_pmove(I16 AxisNo, F64 Dist, F64 SpeedLimit);
I16 _7443_pulser_home_move(I16 AxisNo, I16 HomeType, F64 SpeedLimit);
I16 _7443_set_pulser_ratio(I16 AxisNo, I16 PDV, I16 PMG);
I16 _7443_pulser_r_line2(I16 CardNo,I16 *AxisArray, F64 DistX, F64 DistY,
                         F64 SpeedLimit);
I16 _7443_pulser_r_arc2(I16 CardNo,I16 *AxisArray, F64 OffsetCx, F64
                         OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64
                         MaxVel);
```

Visual Basic(Windows95/NT)

```
B_7443_set_pulser_iptmode (ByVal AxisNo As Integer, ByVal InputMode As
                         Integer, ByVal Inverse As Integer) As Integer
B_7443_pulser_vmove (ByVal AxisNo As Integer, ByVal SpeedLimit As
                         Double) As Integer
B_7443_pulser_pmove (ByVal AxisNo As Integer, ByVal Dist As Double,
                         ByVal SpeedLimit As Double) As Integer
B_7443_pulser_home_move (ByVal AxisNo As Integer, ByVal HomeType As
                         Integer, ByVal SpeedLimit As Double) As Integer
B_7443_set_pulser_ratio (ByVal AxisNo As Integer, ByVal PDV As Integer,
                         ByVal PMG As Integer) As Integer
B_7443_pulser_r_line2 (ByVal CardNo As Integer, AxisArray As Integer,
                         ByVal DistX As Double, Byval DistY As Double,
                         ByVal SpeedLimit As Double) As Integer
B_7443_pulser_r_arc2 (ByVal CardNo As Integer, AxisArray As Integer,
                         ByVal OffsetCx As Double, ByVal Offset As Double,
                         ByVal OffsetEx As Double, ByVal OffsetEy As
                         Double, ByVal DIR As Integer, ByVal MaxVel As
                         Double) As Integer
```

@パラメータ

AxisNo: 軸番号

InputMode: ﾊﾟﾙｻｰ入力ﾓｰﾄﾞの設定
ipt_mode=0： １逓倍のＡＢ相入力
ipt_mode=1： ２逓倍のＡＢ相入力
ipt_mode=2： ４逓倍のＡＢ相入力
ipt_mode=3： CW/CCW 入力

Inverse: 入力ﾊﾟﾙｽの方向と出力ﾊﾟﾙｽの方向を逆に設定
Inverse=0： 逆にしない
inverse=1： 出力ﾊﾟﾙｽを逆に設定

SpeedLimit: ﾊﾟﾙｻｰ動作における最高速度
例えば、SpeedLimit が 100pps に設定されているならば軸は最も速い 100pps で動くことができます。そして、入力ﾊﾟﾙｻｰ信号ﾚｰﾄも 100pps である。

Dist: ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞでの移動距離

HomeType: 原点移動ﾀｲﾌﾟ
HomeType=0： ｺﾏﾝﾄﾞ原点(ｺﾏﾝﾄﾞｶｳﾝﾀが「0」になったら停止)
HomeType=1： 原点信号(ORG)

CardNo: 基板番号

DistX: Ｘ軸の移動距離

DistY: Ｙ軸の移動距離

AxisArray: 補間動作を実行する軸番号の配列
例: Int AxisArray2={0,2}; //軸 0 と 2
Int AxisArray2={1,3}; //軸 1 と 3
注意:AxisArray[0]は AxisArray[1]よりも小さくなければならない。

OffsetCx: 中心のＸ軸ｵﾌｾｯﾄ量

OffsetCy: 中心のＹ軸ｵﾌｾｯﾄ量

OffsetEx: 円弧終点のＸ軸ｵﾌｾｯﾄ量

OffsetEy: 円弧終点のＹ軸ｵﾌｾｯﾄ量

DIR: 円弧の方向
DIR = 0：CW, DIR = 1：CCW

MaxVel: ﾊﾟﾙｻｰ動作における最高速度

@復帰コード

ERR_NoError : 0
ERR_PulserHomeTypeError : 35

6.11　軸の動作状態

@名前

_7443_motion_done　　軸の状態監視

@記述

_7443_motion_done

PPCI7443 の動作状態を返します。

@構文

C/C++(DOS、Windows95/NT)

```
I16 _7443_motion_done(I16 AxisNo);
```

Visual Basic(Windows95/NT)

```
B_7443_motion_done (ByVal AxisNo As Integer) As Integer
```

@パラメータ

AxisNo:　軸番号

@戻り値

0　停止中
1　未使用（常時「0」）
2　未使用（常時「0」）
3　未使用（常時「0」）
4　他の軸の待ち
5　ERC 完了待ち
6　DIR 変更待ち
7　ﾊﾞｯｸﾗｯｼｭ補正中
8　PA/PB 待ち
9　原点への特殊な速度移動中
10　移動開始速度での動作中
11　加速中
12　最高速度で動作中
13　減速中
14　INP 待ち
15　その他（ｽﾀｰﾄ制御中）

6.12 ｻｰﾎﾞﾄﾞﾗｲﾊﾞｲﾝﾀｰﾌｪｰｽ

@名前

_7443_set_alm	ｱﾗｰﾑ論理とｱﾗｰﾑﾓｰﾄﾞの設定
_7443_set_el	EL論理とELﾓｰﾄﾞの設定
_7443_set_inp	INP論理とINPﾓｰﾄﾞの設定
_7443_set_erc	ERC論理と出力時間の設定
_7443_set_servo	SVON信号(汎用出力)のON/OFF
_7443_set_sd	SD論理とSDﾓｰﾄﾞの設定

@記述

_7443_set_alm_logic

ｻｰﾎﾞﾄﾞﾗｲﾊﾞからの ALARM 信号入力の論理を設定します。ALARM 信号が ON したときに、２つ対応するﾓｰﾄﾞが利用可能となります。

_7443_set_el

EL 信号が ON したときの軸の停止ﾓｰﾄﾞを設定します。

_7443_set_inp_logic

ｻｰﾎﾞﾄﾞﾗｲﾊﾞからの INP 信号の論理を設定します。ﾕｰｻﾞｰはこのﾌｧﾝｸｼｮﾝで信号の有効/無効の設定もできます。初期設定は無効になっています。

_7443_set_erc

ERC 信号の論理と出力時間を設定します。

_7443_set_servo

SVON 信号の ON/OFF が制御を行います。初期設定は１(OFF)で、これは SVON 信号が GND とｵｰﾌﾟﾝ状態であることを意味します。

_7443_set_sd_logic

機械的なｼｽﾃﾑからの SD 信号の論理、ﾗｯﾁ制御、および操作ﾓｰﾄﾞを設定します。ﾕｰｻﾞｰは有効/無効の設定ができます。初期設定は無効になっています。

@構文

C/C++(DOS、Windows95/NT)

```
I16 _7443_set_alm(I16 AxisNo, I16 alm_logic, I16 alm_mode);
I16 _7443_set_el(I16 AxisNo, I16 el_mode);
I16 _7443_set_inp(I16 AxisNo, I16 inp_enable, I16 inp_logic);
I16 _7443_set_erc(I16 AxisNo, I16 erc_logic, I16 erc_on_time);
I16 _7443_set_servo(I16 AxisNo, I16 on_off);
I16 _7443_set_sd(I16 AxisNo, I16 enable, I16 sd_logic, I16 sd_latch, I16
                 sd_mode);
```

Visual Basic(Windows95/NT)

```
B_7443_set_alm (ByVal AxisNo As Integer, ByVal alm_logic As Integer,
                ByVal alm_mode As Integer) As Integer
```

B_7443_set_el (ByVal AxisNo As Integer, ByVal el_mode As Integer) As Integer
B_7443_set_inp (ByVal AxisNo As Integer, ByVal inp_enable As Integer, ByVal inp_logic As Integer) As Integer
B_7443_set_erc (ByVal AxisNo As Integer, ByVal erc_logic As Integer, ByVal erc_on_time As Integer) As Integer
B_7443_set_servo (ByVal AxisNo As Integer, ByVal On_Off As Integer) As Integer
B_7443_set_sd (ByVal AxisNo As Integer, ByVal enable As Integer, ByVal sd_logic As Integer, ByVal sd_latch As Integer, ByVal sd_mode As Integer) As Integer

@パラメータ

AxisNo: 軸番号
alm_logic: ALARM 信号の論理
alm_logic=0： Active Low
alm_logic=1： Active High
alm_mode: ALARM 信号入力時の軸停止ﾓｰﾄﾞ
alm_mode=0： 急停止(初期値)
alm_mode=1： 減速停止
el_mode: EL 信号入力時の軸停止ﾓｰﾄﾞ
el_mode=0： 急停止(初期値)
el_mode=1： 減速停止
inp_enable: INP 信号の有効/無効
inp_enable=0：無効(初期値)
inp_enable=1：有効
inp_logic: INP の論理
inp_logic=0： Active LOW
inp_logic=1： Active HIGH
erc_logic: ERC 信号の論理
erc_logic=0： Active Low
erc_logic=1： Active High
erc_on_time: ERC 信号の出力時間
erc_on_time=0 12 us
erc_on_time=1 102 us
erc_on_time=2 409 us
erc_on_time=3 1.6 ms
erc_on_time=4 13 ms
erc_on_time=5 52 ms
erc_on_time=6 104 ms
on_off: SVON 信号の ON-OFF 状態
on_off=0： ON
on_off=1： OFF
enable: SD 信号の有効/無効
enable=0： 無効(初期値)
enable=1： 有効
sd_logic: SD 信号の論理
sd_logic=0： Active Low
sd_logic=1： Active High
sd_latch: SD 信号のﾗｯﾁ制御
sd_latch=0： ﾗｯﾁしない
sd_latch=1： ﾗｯﾁする
sd_mode: SD 信号入力時の軸の対応ﾓｰﾄﾞ
sd_mode=0： 減速のみ
sd_mode=1： 減速して停止

@復帰コード

ERR_NoError : 0

6.13　I/O状態の読み込み

@ 名前

_7443_get_io_status　　PPCI7443のすべてのI/O状態の読み込み

@ 記述

_7443_get_io_status

各軸すべての入出力状態を読み込みます。各ﾋﾞｯﾄの意味は次の表に示します。

ﾋﾞｯﾄ	名前	記述
0	RDY	RDY 信号入力
1	ALM	ｱﾗｰﾑ信号
2	+EL	＋側ｴﾝﾄﾞﾘﾐｯﾄｽｲｯﾁ
3	-EL	－側ｴﾝﾄﾞﾘﾐｯﾄｽｲｯﾁ
4	ORG	原点ｽｲｯﾁ
5	DIR	DIR 信号出力
6	未使用	常時「0」
7	PCS	PCS 信号入力
8	ERC	ERC 信号出力
9	EZ	Ｚ相信号
10	未使用	常時「0」
11	Latch	ﾗｯﾁ信号入力
12	SD	ｽﾛｰﾀﾞｳﾝ信号入力
13	INP	INP 信号入力
14	SVON	ｻｰﾎﾞ ON 信号出力

@ 構文

C/C++ (DOS, Windows 95/98/NT)

```
I16 _7443_get_io_status (I16 AxisNo, U16 *io_sts);
```

Visual Basic (Windows 95/NT)

```
B_7443_get_io_status (ByVal AxisNo As Integer, io_sts As Integer) As
                     Integer
```

@ パラメータ

AxisNo:　軸番号

***io_sts:**　入出力状態

「1」が ON で「0」が OFF を示します。ON/OFF 状態は対応する論理設定に基づいて読み込まれます。

@ 復帰ｺｰﾄﾞ

ERR_NoError　　: 0

6.14　割り込み信号制御

@ 名前

_7443_int_control	INT動作の有効/無効の設定
_7443_set_int_factor	INT要素の設定
_7443_int_enable	割り込みｲﾍﾞﾝﾄを有効に設定(Windowsのみ)
_7443_int_disable	割り込みｲﾍﾞﾝﾄを無効に設定(Windowsのみ)
_7443_get_int_status	INT状態の読み込み(Windowsのみ)
_7443_link_interrupt	割り込みｺｰﾙﾊﾞｯｸﾌｧﾝｸｼｮﾝのﾘﾝｸ設定 (Windowsのみ)
_7443_get_int_type	INTﾀｲﾌﾟの読み込み(Dosのみ)
_7443_enter_isr	割り込みｻｰﾋﾞｽﾙｰﾁﾝに入った通知 (Dosのみ)
_7443_leave_isr	割り込みｻｰﾋﾞｽﾙｰﾁﾝから出た通知 (Dosのみ)
_7443_get_event_int	event_int状態の読み込み (Dosのみ)
_7443_get_error_int	error_int状態の読み込み (Dosのみ)
_7443_get_irq_status	IRQ状態の読み込み status (Dosのみ)
_7443_not_my_irq	指定基板以外のIRQ (Dosのみ)
_7443_isr0~9, a, b	個々の割り込みｻｰﾋﾞｽﾙｰﾁﾝ (Dosのみ)
_7443_set_axis_stop_int	軸の割り込み停止を有効に設定
_7443_mask_axis_stop_int	軸の割り込み停止をﾏｽｸする

@ 記述

_7443_int_control

ﾎｽﾄPCへの割り込み発生を有効に設定するのに使用します。

_7443_set_int_factor

割り込み信号を発生させる要素を割り当てます。

_7443_int_control()ｺﾏﾝﾄﾞが有効に設定されている時、割り当てられた信号がONになると割り込み信号が出力されます。

PPCI7443の割り込み要素には２種類(error_int_statusとevent_int_status)で構成されます。event_int_status が通常の動作のもとでの割り込み要素で、この種の割り込み信号は **_7443_set_int_factor()**ｺﾏﾝﾄﾞによってﾏｽｸすることができます。

error_int_status は PPCI7443 の異常時の割り込み要素です。例えば、ｴﾝﾄﾞﾘﾐｯﾄ、ｱﾗｰﾑなどです。この種の割り込みはﾏｽｸすることができません。下記に２つの割り込み要素の一覧を示します。対応するﾋﾞｯﾄを「1」に設定することにより、PPCI7443はﾎｽﾄPCに割り込み信号を発生させることができます。

ﾋﾞｯﾄ	記述
0	通常の停止時

1	次のコマンドが継続スタート時
2	連続動作用ﾌﾟﾘﾚｼﾞｽﾀ2書き込み可能時
3	（予約）
4	加速開始時
5	加速終了時
6	減速開始時
7	減速終了時
8	（予約）
9	（予約）
10	脱調が発生時
11	汎用ｺﾝﾊﾟﾚｰﾀが比較時
12	軸０，軸１の比較ﾄﾘｶﾞｰ出力時
13	（予約）
14	軸２，軸３のﾗｯﾁ入力時
15	ＯＲＧが入力によるカウント値のラッチ時
16	ＳＤが入力ＯＮ時
17	（予約）
18	（予約）
19	同時スタート入力ＯＮ時
20~31	（予約）

_7443_int_enable *(Windowsのみ)*

Windows の割り込みｲﾍﾞﾝﾄを有効に設定します。

_7443_int_disable *(Windowsのみ)*

Windows の割り込みｲﾍﾞﾝﾄを無効に設定します。

_7443_get_int_status *(Windowsのみ)*

どの割り込み信号が発生されたかを識別するのに使用します。このﾌｧﾝｸｼｮﾝによって値を取得後、ｽﾃｰﾀｽﾚｼﾞｽﾀは「0」にｸﾘｱされます。戻り値は32ﾋﾞｯﾄ符号なし整数が２個です。一つがerror_int_statusです。error_int_statusの対応するﾋﾞｯﾄの一覧を下記の表に示します。

error_int_status ： ﾏｽｸできない	
ﾋﾞｯﾄ	割り込み要素
0	＋側ソフトリミットによる停止
1	－側ソフトリミットによる停止
2	（予約）
3	汎用ｺﾝﾊﾟﾚｰﾀが条件成立による停止
4	（予約）
5	＋側エンドリミットによる停止

6	ー側エンドリミットによる停止
7	アラーム入力による停止
8	CSTP 入力による停止
9	CEMG 入力による停止
10	ＳＤ入力による停止
11	（予約）
12	補間データ異常による停止
13	補間動作中に他軸の異常停止による停止
14	ﾊﾟﾙｻｰ入力用ﾊﾞｯﾌｧｶｳﾝﾀｰのｵｰﾊﾞｰﾌﾛｰによる停止
15	補間動作時の位置決めｶｳﾝﾀのｶｳﾝﾄ範囲ｵｰﾊﾞｰによる停止
16	エンコーダ入力エラー（停止しません）
17	パルサー入力エラー（停止しません）
18~31	（予約）

もう一つが event_int_status です。こちらは
_7443_set_int_factor()ｺﾏﾝﾄﾞでﾏｽｸすることができます。
event_int_status の対応するﾋﾞｯﾄの一覧を下記の表に示します。

event_int_status：*_7443_int_*factor()ｺﾏﾝﾄﾞによりﾏｽｸできる	
ﾋﾞｯﾄ	記述
0	通常の停止時
1	次のコマンドが継続スタート時
2	連続動作用ﾌﾟﾘﾚｼﾞｽﾀ 2 書き込み可能時
3	（予約）
4	加速開始時
5	加速終了時
6	減速開始時
7	減速終了時
8	（予約）
9	（予約）
10	脱調が発生時
11	汎用ｺﾝﾊﾟﾚｰﾀが比較時
12	軸０，軸１の比較ﾄﾘｶﾞｰ出力時
13	（予約）
14	軸２，軸３のﾗｯﾁ入力時
15	ＯＲＧが入力によるカウント値のラッチ時
16	ＳＤが入力ＯＮ時

17	（予約）
18	（予約）
19	同時スタート入力ＯＮ時
20~31	（予約）

_7443_link_interrupt *(Windowsのみ)*
割り込みｺｰﾙﾊﾞｯｸﾌｧﾝｸｼｮﾝをﾘﾝｸさせるのに使用します。

_7443_get_int_type *(Dosのみ)*
どの割り込みの種類が発生したかを検出するのに使用します。

_7443_enter_isr *(Dosのみ)*
今割り込みｻｰﾋﾞｽﾙｰﾁﾝに入った処理を通知するのに使用します。

_7443_leave_isr *(Dosのみ)*
今割り込みｻｰﾋﾞｽﾙｰﾁﾝから出た処理を通知するのに使用します。

_7443_get_event_int *(Dosのみ)*
event_int_status を取得するのに使用します。

_7443_get_error_int *(Dosのみ)*
error_int_status を取得するのに使用します。

_7443_get_irq_status *(Dosのみ)*
指定した基板が割り込み信号をﾎｽﾄ PC に発生しているかどうかを確認することを可能にします。

_7443_not_my_irq *(Dosのみ)*
指定した基板以外が割り込み信号をﾎｽﾄ PC に発生したのを知った後で、呼び出さなくてはなりません。

_7443_isr0, _7443_isr1, _7443_isr2, _7443_isr3, ….. _7443_isr9, _7443_isra, _7443_isrb *(Dosのみ)*
基板０から１１の個々の割り込みｻｰﾋﾞｽﾙｰﾁﾝです。

_7443_set_axis_stop_int
軸の停止割り込みを有効/無効に設定します。有効に設定すると、通常の停止、ｴﾗｰ停止にかかわらず割り込みを発生させます。この割り込み状態は_7443_mask_axis_stop_int()ｺﾏﾝﾄﾞの設定により、ON/OFF させることができます。この割り込みは_7443_set_int_factor()ｺﾏﾝﾄﾞによって設定された要素とは異なります。連続した動作のとき最終のｺﾏﾝﾄﾞが完了したときのみ割り込みを発生させるのに非常に役立ちます。

_7443_mask_axis_stop_int
_7443_set_axis_stop_int()ｺﾏﾝﾄﾞによって設定された割り込み要素をﾏｽｸします。但し、連続動作中の割り込みのみで、最終ｺﾏﾝﾄﾞの停止割り込みはﾏｽｸしません。（連続動作でないときは無意味です）

@ 構文

C/C++ (DOS)

```
I16 _7443_int_control(U16 cardNo, U16 intFlag );
I16 _7443_set_int_factor(I16 AxisNo, U32 int_factor );
I16 _7443_get_int_type(I16 AxisNo, U16 *int_type);
I16 _7443_enter_isr(I16 AxisNo);
I16 _7443_leave_isr(I16 AxisNo);
I16 _7443_get_event_int(I16 AxisNo, U32 *event_int);
I16 _7443_get_error_int(I16 AxisNo, U32 *error_int);
I16 _7443_get_irq_status(U16 cardNo, U16 *sts);
I16 _7443_not_my_irq(I16 CardNo);
void interrupt _7443_isr0 (void);
void interrupt _7443_isr1 (void);
void interrupt _7443_isr2 (void);
void interrupt _7443_isr3 (void);
void interrupt _7443_isr4 (void);
void interrupt _7443_isr5 (void);
void interrupt _7443_isr6 (void);
void interrupt _7443_isr7 (void);
void interrupt _7443_isr8 (void);
void interrupt _7443_isr9 (void);
void interrupt _7443_isra (void);
void interrupt _7443_isrb (void);
```

C/C++ (Windows 95/98/NT)

```
I16 _7443_int_control(U16 cardNo, U16 intFlag );
I16 _7443_set_int_factor(I16 AxisNo, U32 int_factor );
I16 _7443_int_enable(I16 CardNo, HANDLE *phEvent);
I16 _7443_int_disable(I16 CardNo);
I16 _7443_get_int_status(I16 AxisNo, U32 *error_int_status, U32
                         *event_int_status );
I16 _7443_link_interrupt(I16 CardNo, void ( __stdcall *callbackAddr)(I16
                         IntAxisNoInCard));
I16 _7443_set_axis_stop_int(I16 AxisNo, I16 axis_stop_int);
I16 _7443_mask_axis_stop_int(I16 AxisNo, I16 int_disable);
```

Visual Basic (Windows 95/NT)

```
B_7443_int_control (ByVal CardNo As Integer, ByVal intFlag As Integer) As
                   Integer
B_7443_set_int_factor (ByVal AxisNo As Integer, ByVal int_factor As Long)
                   As Integer
B_7443_int_enable (ByVal CardNo As Integer, phEvent As Long) As Integer
B_7443_int_disable (ByVal CardNo As Integer) As Integer
B_7443_get_int_status (ByVal AxisNo As Integer, error_int_status As Long,
                   event_int_status As Long) As Integer
B_7443_link_interrupt (ByVal CardNo As Integer, ByVal lpCallBackProc As
                   Long) As Integer
B_7443_set_axis_stop_int (ByVal AxisNo As Integer, ByVal axis_stop_int As
                   Integer) As Integer
B_7443_mask_axis_stop_int (ByVal AxisNo As Integer, ByVal int_disable As
                   Integer) As Integer
```

@ パラメータ

cardNo: 基板番号
AxisNo: 軸番号
intFlag: 割り込みﾌﾗｸﾞ 0 or 1
intFlag=0： 無効
intFlag=1： 有効
int_factor: 割り込み要素（前の表を参照）
***int_type:** 割り込みﾀｲﾌﾟ
int_type=1： error int
int_type=2： event int
int_type=3： both happened
***event_int:** event_int_status（前の表を参照）
***error_int:** error_int_status（前の表を参照）
***sts:** sts=0： 指定した基板以外の IRQ
sts=1： 指定した基板の IRQ
***phEvent:** ｲﾍﾞﾝﾄﾊﾝﾄﾞﾗ (Windows)
***error_int_status**: 前の表を参照
***event_int_status:** 前の表を参照
axis_stop_int:停止割り込みの有効/無効
axis_stop_int=0：無効
axis_stop_int=1：有効
int_disable: 停止割り込みのﾏｽｸ
int_disable=0： ﾏｽｸしない
int_disable=1： ﾏｽｸする

@ 復帰ｺｰﾄﾞ

ERR_NoError : 0
ERR_EventNotEnableYet : 37
ERR_LinkIntError : 31
ERR_CardNoError : 30

6.15　位置制御とｶｳﾝﾀ

@ 名前

_7443_get_position	ﾌｨｰﾄﾞﾊﾞｯｸﾎﾟｼﾞｼｮﾝｶｳﾝﾀの値の読み込み
_7443_set_position	ﾌｨｰﾄﾞﾊﾞｯｸﾎﾟｼﾞｼｮﾝｶｳﾝﾀの値の設定
_7443_get_command	ｺﾏﾝﾄﾞﾎﾟｼﾞｼｮﾝｶｳﾝﾀの値の読み込み
_7443_set_command	ｺﾏﾝﾄﾞﾎﾟｼﾞｼｮﾝｶｳﾝﾀの値の設定
_7443_get_error_counter	ﾎﾟｼﾞｼｮﾝｴﾗｰｶｳﾝﾀの値の読み込み
_7443_reset_error_counter	ﾎﾟｼﾞｼｮﾝｴﾗｰｶｳﾝﾀの値のﾘｾｯﾄ
_7443_get_general_counter	汎用ｶｳﾝﾀの値の読み込み
_7443_set_general_counter	汎用ｶｳﾝﾀの値の設定
_7443_get_target_pos	ﾀｰｹﾞｯﾄﾎﾟｼﾞｼｮﾝﾚｺｰﾀﾞの値の読み込み
_7443_reset_target_pos	ﾀｰｹﾞｯﾄﾎﾟｼﾞｼｮﾝﾚｺｰﾀﾞの値のﾘｾｯﾄ
_7443_get_rest_command	動作完了までの残りﾊﾟﾙｽの読み込み
_7443_check_rdp	ｽﾛｰﾀﾞｳﾝﾎﾟｲﾝﾄﾃﾞｰﾀの読み込み

@ 記述

_7443_get_position()

ﾌｨｰﾄﾞﾊﾞｯｸﾎﾟｼﾞｼｮﾝｶｳﾝﾀの値を読み込むのに使用します。注意、この値は、動作比率によって処理された値です。もし動作比率が 0.5 の場合、読み込まれた値はｶｳﾝﾀ値の２倍となります。ﾌｨｰﾄﾞﾊﾞｯｸｶｳﾝﾀのｶｳﾝﾀ入力は***_7443_set_feedback_src()***ｺﾏﾝﾄﾞによって外部からの EA/EB 信号か PPCI7443 の出力ﾊﾟﾙｽのどちらかが選択されます。

_7443_set_position()

ﾌｨｰﾄﾞﾊﾞｯｸﾎﾟｼﾞｼｮﾝｶｳﾝﾀの値を指定した値に変更するのに使用します。注意、設定される値は動作比率によって処理されます。もし動作比率が 0.5 の場合、設定される値は、与えられる値の２倍となります。

_7443_get_command()

ｺﾏﾝﾄﾞﾎﾟｼﾞｼｮﾝｶｳﾝﾀの値を読み込むのに使用します。ｺﾏﾝﾄﾞﾎﾟｼﾞｼｮﾝｶｳﾝﾀのｶｳﾝﾀ入力は PPCI7443 の出力ﾊﾟﾙｽです。

_7443_set_command()

ｺﾏﾝﾄﾞﾎﾟｼﾞｼｮﾝｶｳﾝﾀの値を変更するのに使用します。

_7443_get_error_counter()

ﾎﾟｼﾞｼｮﾝｴﾗｰｶｳﾝﾀの値を読み込むのに使用します。

_7443_reset_error_counter()

ﾎﾟｼﾞｼｮﾝｴﾗｰｶｳﾝﾀの値をｸﾘｱするのに使用します。

_7443_get_general_counter()

汎用ｶｳﾝﾀの値を読み込むのに使用します。

_7443_set_general_counter()

汎用ｶｳﾝﾀの値の変更とｶｳﾝﾀ入力を設定するのに使用します。（初期設定はﾊﾟﾙｻｰ入力です）

_7443_get_target_pos()

ﾀｰｹﾞｯﾄﾎﾟｼﾞｼｮﾝﾚｺｰﾀﾞの値を読み込むのに使用します。ﾀｰｹﾞｯﾄﾎﾟｼﾞｼｮﾝﾚｺｰﾀﾞの値はPPCI7443 ｿﾌﾄｳｪｱﾄﾞﾗｲﾊﾞによって保持されます。それは、現在動作中である軸を止める位置を記録します。

_7443_reset_target_pos()

ﾀｰｹﾞｯﾄﾎﾟｼﾞｼｮﾝﾚｺｰﾀﾞに新しい値を設定するのに使用します。原点復帰完了か***_7443_set_position()***ｺﾏﾝﾄﾞによって新しいﾌｨｰﾄﾞﾊﾞｯｸｶｳﾝﾀの値が設定されたとき、このﾌｧﾝｸｼｮﾝを呼び出すことが必要です。

_7443_get_rest_command()

現在動作中の軸が停止するまでの残りﾊﾟﾙｽ数を読み込むのに使用します。

_7443_check_rdp()

ｽﾛｰﾀﾞｳﾝﾎﾟｲﾝﾄﾃﾞｰﾀを読み込むのに使用します。ｽﾛｰﾀﾞｳﾝﾎﾟｲﾝﾄとは、減速が始まる位置のことです。ﾃﾞｰﾀはﾊﾟﾙｽ数として記憶され、移動中の軸の残りﾊﾟﾙｽがこのﾃﾞｰﾀに達したとき、減速させます。

@ 構文

C/C++ (DOS, Windows 95/98/NT)

```
I16 _7443_get_position(I16 AxisNo, F64 *pos);
I16 _7443_set_position(I16 AxisNo, F64 pos);
I16 _7443_get_command(I16 AxisNo, I32 *cmd);
I16 _7443_set_command(I16 AxisNo, I32 cmd);
I16 _7443_get_error_counter(I16 AxisNo, I16 *error_counter);
I16 _7443_reset_error_counter(I16 AxisNo);
I16 _7443_get_general_counter(I16 AxisNo, F64 *CntValue);
I16 _7443_set_general_counter(I16 AxisNo,I16 CntSrc, F64 CntValue);
I16 _7443_get_target_pos(I16 AxisNo, F64 *T_pos);
I16 _7443_reset_target_pos(I16 AxisNo, F64 T_pos);
I16 _7443_get_rest_command(I16 AxisNo, I32 *rest_command);
I16 _7443_check_rdp(I16 AxisNo, I32 *rdp_command);
```

Visual Basic (Windows 95/NT)

```
B_7443_get_position (ByVal AxisNo As Integer, Pos As Double) As Integer
B_7443_set_position (ByVal AxisNo As Integer, ByVal Pos As Double) As
                    Integer
B_7443_get_command (ByVal AxisNo As Integer, cmd As Long) As Integer
B_7443_set_command (ByVal AxisNo As Integer, ByVal cmd As Long) As
                    Integer
B_7443_get_error_counter (ByVal AxisNo As Integer, error_counter As
                    Integer) As Integer
B_7443_reset_error_counter (ByVal AxisNo As Integer) As Integer
B_7443_get_general_counter (ByVal AxisNo As Integer, CntValue As
                    Double) As Integer
```

B_7443_set_general_counter (ByVal AxisNo As Integer, ByVal CntSrc As Integer, ByVal CntValue As Double) As Integer
B_7443_get_target_pos (ByVal AxisNo As Integer, Pos As Double) As Integer
B_7443_reset_target_pos (ByVal AxisNo As Integer, ByVal Pos As Double) As Integer
B_7443_get_rest_command (ByVal AxisNo As Integer, rest_command As Long) As Integer
B_7443_check_rdp (ByVal AxisNo As Integer, rdp_command As Long) As Integer

@ パラメータ

AxisNo: 軸番号
Pos,*Pos: ﾌｨｰﾄﾞﾊﾞｯｸﾎﾟｼﾞｼｮﾝｶｳﾝﾀの値
(範囲： -134217728 ～ 134217727)
cmd,*cmd: ｺﾏﾝﾄﾞﾎﾟｼﾞｼｮﾝｶｳﾝﾀの値
(範囲： -134217728 ～ 134217727)
error_counter,*error_counter: ﾎﾟｼﾞｼｮﾝｴﾗｰｶｳﾝﾀの値
(範囲： -32768 ～ 32767)
T_pos,*T_pos: ﾀｰｹﾞｯﾄﾎﾟｼﾞｼｮﾝﾚｺｰﾀﾞの値
(範囲： -134217728 ～ 134217727)
CntValue,*CntValue: 汎用ｶｳﾝﾀの値
(範囲： -134217728 ～ 134217727)
rest_command,*rest_command: 軸終了までの残りﾊﾟﾙｽ数
(範囲： -134217728 ～ 134217727)
rdp_command,*rdp_command: ｽﾛｰﾀﾞｳﾝﾎﾟｲﾝﾄﾃﾞｰﾀ
(範囲： 0 ～ 167777215)
CntSrc: 汎用ｶｳﾝﾀのｶｳﾝﾀ入力
CntSrc=0： ｺﾏﾝﾄﾞﾊﾟﾙｽ
CntSrc=1： EA/EB
CntSrc=2： PA/PB (初期値)
CntSrc=3： CLK÷2

@ 復帰ｺｰﾄﾞ

ERR_NoError : 0
ERR_PosOutofRange : 13

6.16　位置の比較とﾗｯﾁ

@ 名前

_7443_set_ltc_logic	LTC信号の論理設定
_7443_get_latch_data	ﾗｯﾁｶｳﾝﾀﾃﾞｰﾀの読み込み
_7443_set_soft_limit	ｿﾌﾄﾘﾐｯﾄの設定
_7443_enable_soft_limit	ｿﾌﾄﾘﾐｯﾄ機能の有効設定
_7443_disable_soft_limit	ｿﾌﾄﾘﾐｯﾄ機能の無効設定
_7443_set_error_counter_check	脱調検出の設定
_7443_set_general_comparator	汎用ｺﾝﾊﾟﾚｰﾀの設定
_7443_set_trigger_comparator	ﾄﾘｶﾞｰｺﾝﾊﾟﾚｰﾀの設定
_7443_set_trigger_type	ﾄﾘｶﾞｰ出力ﾀｲﾌﾟの設定
_7443_check_compare_data	現在のｺﾝﾊﾟﾚｰﾀﾃﾞｰﾀのﾁｪｯｸ
_7443_check_compare_status	現在のｺﾝﾊﾟﾚｰﾀ状態のﾁｪｯｸ
_7443_set_auto_compare	自動読み込みのためのｺﾝﾍﾟｱﾃﾞｰﾀ源の設定
_7443_build_compare_function	一定間隔でｺﾝﾍﾟｱﾃﾞｰﾀを作る
_7443_build_compare_table	ｺﾝﾍﾟｱﾃｰﾌﾞﾙでｺﾝﾍﾟｱﾃﾞｰﾀを作る
_7443_cmp_v_change	ｺﾝﾊﾟﾚｰﾀによる速度変更
_7443_set_enable_inp	ｶｳﾝﾀをﾗｯﾁする信号の選択

@ 記述

_7443_set_ltc_logic()

ﾗｯﾁ入力信号(LTC)の論理を設定するのに使用します。このﾌｧﾝｸｼｮﾝは、PPCI7443 の後ろの２軸のみに適応されます。

_7443_get_latch_data()

ﾗｯﾁ信号が入力された時の、ｶｳﾝﾀの値を読み込むのに使用します。

_7443_set_soft_limit()

ｿﾌﾄﾘﾐｯﾄの値を設定するのに使用します。

_7443_enable_soft_limit(),_7443_disable_soft_limit()

ｿﾌﾄﾘﾐｯﾄ機能の有効/無効の設定に使用します。 有効に設定すると、ｿﾌﾄﾘﾐｯﾄは機械的なﾘﾐｯﾄｽｲｯﾁと同様に動作します。

_7443_set_error_counter_check()

脱調をﾁｪｯｸする機能を有効に設定するのに使用します。 許容値を与えることにより、ﾎﾟｼﾞｼｮﾝｴﾗｰｶｳﾝﾀが許容値を超えるとき PPCI7443 は割り込み信号(event_int_status のﾋﾞｯﾄ 10)を発生させます。

_7443_set_general_comparator()

汎用ｺﾝﾊﾟﾚｰﾀの比較ﾃﾞｰﾀと比較源の設定に使用します。 ｶｳﾝﾀ値が比較ﾃﾞｰﾀに達したとき PPCI7443 は割り込み信号(event_int_status のﾋﾞｯﾄ 11)を発生させます。

_7443_set_trigger_comparator()

ﾄﾘｶﾞｰｺﾝﾊﾟﾚｰﾀの値と比較方法を設定するのに使用します。ﾌｨｰﾄﾞﾊﾞｯｸﾎﾟｼﾞｼｮﾝｶｳﾝﾀの値が比較値に達したとき、PPCI7443 は CMP ﾋﾟﾝよりﾄﾘｶﾞｰﾊﾟﾙｽを発生し、割り込み信号(event_int_status のﾋﾞｯﾄ 12)もﾎｽﾄ PC に送られます。もし***_7443_set_auto_compare()***ｺﾏﾝﾄﾞが使用されているならば、このﾌｧﾝｸｼｮﾝによって設定された値は無視されます。

注意：このﾌｧﾝｸｼｮﾝは、PPCI7443 の最初の２軸のみに適応されます。

_7443_set_trigger_type()

ﾄﾘｶﾞｰ出力ﾓｰﾄﾞの設定をするのに使用します。

_7443_check_compare_data()

指定されたｺﾝﾊﾟﾚｰﾀの現在の比較ﾃﾞｰﾀを読み込むのに使用します。

_7443_check_compare_status()

すべてのｺﾝﾊﾟﾚｰﾀの状態を読み込むのに使用します。 いづれかのｺﾝﾊﾟﾚｰﾀが成立するとき、***cmp_sts***の対応するﾋﾞｯﾄが「1」、もしくは「0」になります。

_7443_set_auto_compare()

ﾄﾘｶﾞｰｺﾝﾊﾟﾚｰﾀの比較ﾃﾞｰﾀ源を設定するのに使用します。 比較ﾃﾞｰﾀ源はｺﾏﾝﾄﾞ機能か比較ﾃｰﾌﾞﾙのどちらかです。

_7443_build_compare_function()

最初と最後のﾎﾟｲﾝﾄと間隔を定義することにより、比較関数を形成するのに使用します。比較ﾃﾞｰﾀの最高個数には制限がありません。この機能は、比較する最終ﾎﾟｲﾝﾄの後に自動的にﾀﾞﾐｰのﾎﾟｲﾝﾄを設定します。それは、(最終ﾎﾟｲﾝﾄ＋ｲﾝﾀｰﾊﾞﾙ×ﾄｰﾀﾙﾎﾟｲﾝﾄ) ×Move ratio です。

注意：ﾄﾘｶﾞｰ出力中はすべての割り込み機能を OFF に設定して下さい。

_7443_build_compare_table()

ﾃﾞｰﾀ配列を定義することにより、比較ﾃｰﾌﾞﾙを形成するのに使用します。配列の大きさは最大 1024 点です。

注意：ﾄﾘｶﾞｰ出力中はすべての割り込み機能を OFF に設定して下さい。

_7443_cmp_v_change()

ｺﾝﾊﾟﾚｰﾀ速度変更機能のｾｯﾄｱｯﾌﾟに使用します。これは、実際の動作は、V_change ｺﾏﾝﾄﾞですが、汎用ｺﾝﾊﾟﾚｰﾀが成立したときに作用します。このﾌｧﾝｸｼｮﾝが発行されたとき、***_7443_set_general_comparator()*** ｺﾏﾝﾄﾞのﾊﾟﾗﾒｰﾀ ***CmpAction*** は「3」に設定されなけれななりません。比較ﾃﾞｰﾀは***_7443_set_general_comparator()***ｺﾏﾝﾄﾞによって同様に設定されます。残り距離、比較位置速度、新しい速度そして加速時間は ***_7443_cmp_v_change()***ｺﾏﾝﾄﾞにより設定されます。

_7443_enable_inp()

ｶｳﾝﾀをﾗｯﾁする信号を選択します。

@ 構文

C/C++ (DOS, Windows 95/98/NT)

I16 _7443_set_ltc_logic(I16 AxisNo_2or3, I16 ltc_logic);
I16 _7443_get_latch_data(I16 AxisNo, I16 LatchNo, F64 *Pos);
I16 _7443_set_soft_limit(I16 AxisNo, I32 PLimit, I32 NLimit);
I16 _7443_disable_soft_limit(I16 AxisNo);
I16 _7443_enable_soft_limit(I16 AxisNo, I16 Action);
I16 _7443_set_error_counter_check(I16 AxisNo, I16 Tolerance, I16 On_Off);
I16 _7443_set_general_comparator(I16 AxisNo, I16 CmpSrc, I16 CmpMethod, I16 CmpAction, F64 Data);
I16 _7443_set_trigger_comparator(I16 AxisNo,I16 CmpSrc, I16 CmpMethod, F64 Data);
I16 _7443_set_trigger_type(I16 AxisNo, I16 TriggerType);
I16 _7443_check_compare_data(I16 AxisNo, I16 CompType, F64 *Pos);
I16 _7443_check_compare_status(I16 AxisNo, U16 *cmp_sts);
I16 _7443_set_auto_compare(I16 AxisNo ,I16 SelectSrc);
I16 _7443_cmp_v_change(I16 AxisNo, F64 Res_dist, F64 oldvel, F64 newvel, F64 AccTime)
I16 _7443_enable_inp(I16 AxisNo, I16 inp_enable)

C/C++ (Windows 95/98/NT)

I16 _7443_build_compare_function(I16 AxisNo, F64 Start, F64 End, F64 Interval, I16 Device);
I16 _7443_build_compare_table(I16 AxisNo, F64 *TableArray, I32 Size, I16 Device);

C/C++ (Dos)

I16 _7443_build_compare_function(I16 AxisNo, F64 Start, F64 End, F64 Interval);
I16 _7443_build_compare_table(I16 AxisNo, F64 *TableArray, I16 Size);

Visual Basic (Windows 95/NT)

B_7443_set_ltc_logic (ByVal AxisNo As Integer, ByVal ltc_logic As Integer) As Integer
B_7443_get_latch_data (ByVal AxisNo As Integer, ByVal Counter As Integer, Pos As Double) As Integer
B_7443_set_soft_limit (ByVal AxisNo As Integer, ByVal PLimit As Long, ByVal NLimit As Long) As Integer
B_7443_disable_soft_limit (ByVal AxisNo As Integer) As Integer
B_7443_enable_soft_limit (ByVal AxisNo As Integer, ByVal Action As Integer) As Integer
B_7443_set_error_counter_check (ByVal AxisNo As Integer, ByVal Tolerance As Integer, ByVal On_Off As Integer) As Integer
B_7443_set_general_comparator (ByVal AxisNo As Integer, ByVal CmpSrc As Integer, ByVal CmpMethod As Integer, ByVal CmpAction As Integer, ByVal Data As Double) As Integer
B_7443_set_trigger_comparator (ByVal AxisNo As Integer, Byval CmpSrc As Integer, ByVal CmpMethod As Integer, ByVal Data As Double) As Integer

B_7443_set_trigger_type (ByVal AxisNo As Integer, ByVal TriggerType As Integer) As Integer
B_7443_check_compare_data (ByVal AxisNo As Integer, ByVal Cmptype As Integer, Pos As Double) As Integer
B_7443_check_compare_status (ByVal AxisNo As Integer, cmp_sts As Integer) As Integer
B_7443_set_auto_compare (ByVal AxisNo As Integer, ByVal SelectSrc As Integer) As Integer
B_7443_build_compare_function (ByVal AxisNo As Integer, ByVal Start As Double, ByVal End As Double, ByVal Interval As Double, ByVal Device As Integer) As Integer
B_7443_build_compare_table (ByVal AxisNo As Integer, TableArray As Double, ByVal Size As Long, ByVal Device As Integer) As Integer
B_7443_cmp_v_change(ByVal AxisNo As Integer, ByVal Res_dist As Double, ByVal oldvel As Double, ByVal newvel As Double, ByVal AccTime As Double)
B_7443_enable_inp(ByVal AxisNo As Integer, ByVal inp_enable As Integer)

@ パラメータ

AxisNo_2or3: 軸番号（基板の最後の２軸）
ltc_logic: LTC 信号の論理
ltc_logic=0： Active Low
ltc_logic=1： Active High
AxisNo: 軸番号
LatchNo(Counter):ｶｳﾝﾀの指定
Counter=0： ｺﾏﾝﾄﾞｶｳﾝﾀ
Counter=1： ﾌｨｰﾄﾞﾊﾞｯｸｶｳﾝﾀ
Counter=2： ﾎﾟｼﾞｼｮﾝｴﾗｰｶｳﾝﾀ
Counter=3： 汎用ｶｳﾝﾀ
Pos: ﾗｯﾁされたｶｳﾝﾀ値
PLimit: ＋側ｿﾌﾄﾘﾐｯﾄ値
NLimit: －側ｿﾌﾄﾘﾐｯﾄ値
Action: ｿﾌﾄﾘﾐｯﾄ信号 ON 時の処理
Action=0： 割り込みのみ
Action=1： 急停止
Action=2： 減速停止
Action=3： 予約
Tolerance: 脱調検出の許容値
On_Off: 脱調検出の有効/無効の設定
On_Off=0： 無効
On_Off=1： 有効
CmpSrc: 比較源のｶｳﾝﾀ
CmpSrc =0： ｺﾏﾝﾄﾞｶｳﾝﾀ
CmpSrc =1： ﾌｨｰﾄﾞﾊﾞｯｸｶｳﾝﾀ
CmpSrc =2： ﾎﾟｼﾞｼｮﾝｴﾗｰｶｳﾝﾀ
CmpSrc =3： 汎用ｶｳﾝﾀ

CmpMethod: 比較方法
CmpMethod=0： 比較なし
CmpMethod=1： 比較値 = ｶｳﾝﾀ (方向なし)
CmpMethod=2： 比較値 = ｶｳﾝﾀ (＋Dir)
CmpMethod=3： 比較値 = ｶｳﾝﾀ (－Dir)
CmpMethod=4： 比較値 > ｶｳﾝﾀ
CmpMethod=5： 比較値 < ｶｳﾝﾀ

CmpAction: 比較成立時の処理
CmpAction=0： INT のみ
CmpAction=1： 急停止
CmpAction=2： 減速停止
CmpAction=3： 速度変更

Data: 比較値

TriggerType: ﾄﾘｶﾞｰ出力ﾓｰﾄﾞのﾀｲﾌﾟ選択
TriggerType=0： ﾉｰﾏﾙﾊｲﾚﾍﾞﾙ（初期値）
TriggerType=1： ﾉｰﾏﾙﾛｰﾚﾍﾞﾙ

cmp_sts: ｺﾝﾊﾟﾚｰﾀの状態
Bit：0 ＋側ｿﾌﾄﾘﾐｯﾄ ON
Bit：1 －側ｿﾌﾄﾘﾐｯﾄ ON
Bit：2 ｴﾗｰｶｳﾝﾀｺﾝﾊﾟﾚｰﾀ ON
Bit：3 汎用ｺﾝﾊﾟﾚｰﾀ ON
Bit：4 ﾄﾘｶﾞｰｺﾝﾊﾟﾚｰﾀ ON（軸０， １のみ）

SelectSrc: 比較ﾃﾞｰﾀ源
SelectSrc=0： 自動比較無効
SelectSrc=1： FIFO 使用

Start: 比較機能開始位置

End: 比較機能終了位置

Interval: 比較機能間隔

TableArray: 比較ﾃﾞｰﾀ配列

Size: ﾃｰﾌﾞﾙ配列の大きさ

Device: 比較ﾃﾞｰﾀを再ﾛｰﾄﾞするﾃﾞﾊﾞｲｽの選択
Device=0： RAM と割り込み (Windows のみ)
Device=1： FIFO

Res_dist: 比較位置からの残り距離
比較後、最初の目標位置は無視され、軸は Res_dist を移動させます。

oldvel: 比較位置での速度（手動で指定する）

newvel: 新しい速度

AccTime: 加速時間

CompType: ｺﾝﾊﾟﾚｰﾀの状態
CompType=1： ＋側ｿﾌﾄﾘﾐｯﾄの値
CompType=2： －側ｿﾌﾄﾘﾐｯﾄの値
CompType=3： ｴﾗｰｶｳﾝﾀｺﾝﾊﾟﾚｰﾀの値
CompType=4： 汎用ｺﾝﾊﾟﾚｰﾀの値
CompType=5： ﾄﾘｶﾞｰｺﾝﾊﾟﾚｰﾀ出力の値

lnp_enable: ｶｳﾝﾀをﾗｯﾁする信号

lnp_enable=0：	LTCﾋﾟﾝ
lnp_enable=1：	ORGﾋﾟﾝ
lnp_enable=2：	ｶｳﾝﾀ 4 が ON
lnp_enable=3：	ｶｳﾝﾀ 5 が ON

@ 復帰ｺｰﾄﾞ

ERR_NoError	： 0
ERR_CompareNoError	: 18
ERR_CompareMethodError	: 17
ERR_CompareAxisError	: 19
ERR_CompareTableSizeError	: 20
ERR_CompareFunctionError	: 21
ERR_CompareTableNotReady	: 22
ERR_CompareLineNotReady	: 23
ERR_HardwareCompareAxisWrong	: 32
ERR_AutocompareSourceWrong	: 33
ERR_CompareDeviceTypeError	: 34

6.17　連続動作

@ 名前

_7443_set_continuous_move　　連続動作の有効/無効設定
_7443_check_continuous_buffer　ﾌﾟﾘﾚｼﾞｽﾀの空きﾁｪｯｸ

@ 記述

_7443_set_continuous_move()

このﾌｧﾝｸｼｮﾝは、連続する動作の前後で必要です。

_7443_check_continuous_buffer()

ｺﾏﾝﾄﾞﾊﾞｯﾌｧが空であるかどうかを検出するのに使用します。ｺﾏﾝﾄﾞﾊﾞｯﾌｧが空の場合、ﾕｰｻﾞｰは次の動作ｺﾏﾝﾄﾞを書き込めます。さもなければ（空でない場合）新しいｺﾏﾝﾄﾞは無視され動作しません。しかし、ﾀｰｹﾞｯﾄﾎﾟｼﾞｼｮﾝｶｳﾝﾀｰのみ変更されますので、ｺﾏﾝﾄﾞﾎﾟｼﾞｼｮﾝ、ﾌｨｰﾄﾞﾊﾞｯｸﾎﾟｼﾞｼｮﾝとﾀｰｹﾞｯﾄﾎﾟｼﾞｼｮﾝとで一致しなくなります。

@ 構文

C/C++ (DOS, Windows 95/NT)

```
I16 _7443_set_continuous_move(I16 AxisNo, I16 conti_flag);
I16 _7443_check_continuous_buffer(I16 AxisNo);
```

Visual Basic (Windows 95/NT)

```
B_7443_set_continuous_move (ByVal AxisNo As Integer, ByVal conti_flag
                            As Integer) As Integer
B_7443_check_continuous_buffer (ByVal AxisNo As Integer) As Integer
```

@ パラメータ

AxisNo:　軸番号
conti_flag:　軸動作ﾌﾗｸﾞ
conti_flag=0 :　１回の動作（連続動作終了）
conti_flag=1 :　連続動作　（連続動作開始）

@ 戻り値

ERR_NoError　: 0

_7443_check_continuous_buffer()ｺﾏﾝﾄﾞの戻り値:
ﾊｰﾄﾞｳｪｱﾊﾞｰｼﾞｮﾝのﾋﾞｯﾄ 12 が０の時
0:連続 動作用ﾌﾟﾘﾚｼﾞｽﾀ２は空き
1: 連続動作用ﾌﾟﾘﾚｼﾞｽﾀ２は使用
ﾊｰﾄﾞｳｪｱﾊﾞｰｼﾞｮﾝのﾋﾞｯﾄ 12 が１の時
0: 動作用ﾌﾟﾘﾚｼﾞｽﾀは全て空き
1: 動作用ﾌﾟﾘﾚｼﾞｽﾀ１と２が空き
2: 動作用ﾌﾟﾘﾚｼﾞｽﾀ２が空き
3: 動作用ﾌﾟﾘﾚｼﾞｽﾀは満杯

6.18　複数軸の同時操作

@ 名前

_7443_set_tr_move_all	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞの台形動作での同時操作設定
_7443_set_ta_move_all	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞの台形動作での同時操作設定
_7443_set_sr_move_all	ｲﾝｸﾘﾒﾝﾀﾙﾓｰﾄﾞのＳ字曲線動作での同時操作設定
_7443_set_sa_move_all	ｱﾌﾞｿﾘｭｰﾄﾓｰﾄﾞのＳ字曲線動作での同時操作設定
_7443_start_move_all	複数軸の同時動作の実行
_7443_stop_move_all	複数軸の同時停止
_7443_set_sync_option	他軸との同期設定
_7443_set_sync_stop_mode	同時停止信号による停止ﾓｰﾄﾞの設定

@ 記述

違う基板の複数軸を同時に動作させるのに使用します。複数軸の同時操作とは、指定した複数軸を同時に動作開始させるか停止させることです。動作させる軸は設定ｺﾏﾝﾄﾞのﾊﾟﾗﾒｰﾀ **AxisArray** によって指定され、動作させる軸数はﾊﾟﾗﾒｰﾀ ***TotalAxes*** によって定義されます。

設定ｺﾏﾝﾄﾞによって適切に設定されたとき、***_7443_start_move_all()***ｺﾏﾝﾄﾞは指定された複数の軸を指定された距離だけ動作させます。そして***_7443_stop_move_all()***ｺﾏﾝﾄﾞで同時に停止させます。両方のﾌｧﾝｸｼｮﾝは指定したすべての軸が同時に開始/停止することができます。
注意、もしこれら２つのﾌｧﾝｸｼｮﾝを使用する必要があるならば、ｾｸｼｮﾝ 3.14 を参照して CN4 を配線する必要があります。

下記のﾌﾟﾛｸﾞﾗﾑｺｰﾄﾞはこれらのﾌｧﾝｸｼｮﾝの使用方法の例です。このｺｰﾄﾞは軸０と軸４をそれぞれ 8000.0 と 120000.0 の距離だけ移動させます。もし距離の比率に比例した速度と加速を選択するならば、軸は同時に終了位置に到達するでしょう。（同時動作）

```
int main()
{
                    I16     axes[2] = {0, 4};
                    F64
                            dist[2] = {8000.0, 12000.0},
                            str_vel[2]={0.0, 0.0},
                            max_vel[2]={4000.0, 6000.0},
                            Tacc[2]={0.04, 0.06},
                            Tdec[2]= {0.04, 0.06};
```

```
_7443_set_tr_move_all(2, axes, dist, str_vel, max_vel, Tacc, Tdec);
_7443_start_move_all(axes[0]);
return                  ERR_NoError;
}
```

_7443_set_sync_option()

２個もしくはそれ以上の異なるｺﾏﾝﾄﾞｸﾞﾙｰﾌﾟを同時にｽﾀｰﾄさせます。例えば、２軸の直線補間と１軸の動作を同時に開始させたい場合に、このｺﾏﾝﾄﾞ設定することで実行できます。また、このｺﾏﾝﾄﾞのもう一つの機能は他のｺﾏﾝﾄﾞの完了を待ってｽﾀｰﾄさせることができます。例えば、軸１は軸２の移動完了でｽﾀｰﾄしなければならないときにこのｺﾏﾝﾄﾞを設定することで行えます。

_7443_set_sync_stop_option()

このﾌｧﾝｸｼｮﾝは停止ﾓｰﾄﾞを設定します。停止ﾓｰﾄﾞには２個あり、一つは、急停止でもう一つが減速停止です。_7443_stop_move_all()ｺﾏﾝﾄﾞかもしくは CSTOP 信号が入力されたとき、この設定に従って停止します。

@ 構文

C/C++ (DOS, Windows 95/NT)

```
I16 _7443_set_tr_move_all(I16 TotalAxes, I16 *AxisArray, F64 *DistA, F64
                          *StrVelA, F64 *MaxVelA, F64 *TaccA, F64 *TdecA);
I16 _7443_set_ta_move_all(I16 TotalAxes, I16 *AxisArray, F64 *PosA, F64
                          *StrVelA, F64 *MaxVelA, F64 *TaccA, F64 *TdecA);
I16 _7443_set_sr_move_all(I16 TotalAxes, I16 *AxisArray, F64 *PosA, F64
                          *StrVelA, F64 *MaxVelA, F64 *TaccA, F64 *TdecA,
                          F64 *SVaccA, F64 *SVdecA);
I16 _7443_set_sa_move_all(I16 TotalAxes, I16 *AxisArray, F64 *DistA, F64
                          *StrVelA, F64 *MaxVelA, F64 *TaccA, F64 *TdecA,
                          F64 *SVaccA, F64 *SVdecA);
I16 _7443_start_move_all(I16 FirstAxisNo);
I16 _7443_stop_move_all(I16 FirstAxisNo);
I16 _7443_set_sync_option(I16 AxisNo, I16 sync_stop_on, I16
                          cstop_output_on, I16 sync_option1, I16
                          sync_option2);
I16 _7443_set_sync_stop_mode(I16 AxisNo, I16 stop_mode);
```

Visual Basic (Windows 95/NT)

```
B_7443_set_tr_move_all(ByVal TotalAxes As Integer, AxisArray As Integer,
                       DistA As Double, StrVelA As Double, MaxVelA As
                       Double, TaccA As Double, TdecA As Double)As
                       Integer
B_7443_set_ta_move_all(ByVal TotalAxes As Integer, AxisArray As Integer,
                       PosA As Double, StrVelA As Double, MaxVelA As
                       Double, TaccA As Double, TdecA As double)As
                       Integer
```

B_7443_set_sr_move_all(ByVal TotalAxes As Integer, AxisArray As Integer, DistA As Double, StrVelA As Double, MaxVelA As Double, TaccA As double, TdecA As Double, SVaccA As Double, SVdec As Double)As Integer
B_7443_set_sa_move_all(ByVal TotalAxes As Integer, AxisArray As Integer, PosA As Double, StrVelA As double, MaxVelA As double, TaccA As double, TdecA As double, SVacc As Double, SVdec As Double)As Integer
B_7443_start_move_all(ByVal FirstAxisNo As Integer)As Integer
B_7443_stop_move_all(ByVal FirstAxisNo As Integer)As Integer
B_7443_set_sync_option (ByVal AxisNo As Integer, ByVal sync_stop_on As Integer, ByVal cstop_output_on As Integer, ByVal sync_option1 As Integer, ByVal sync_option2 As Integer) As Integer
B_7443_set_sync_stop_mode (ByVal AxisNo As Integer, ByVal stop_mode As Integer) As Integer

@ パラメータ

TotalAxes: 総軸数 (1 ～ 48)
*** AxisArray**: 軸番号の配列
*** PosA**: ｱﾌﾞｿﾘｭｰﾄ絶対位置の配列 (ﾊﾟﾙｽ数)
*** DistA**: 距離の配列 (ﾊﾟﾙｽ数)
*** StrVelA:** 移動開始速度の配列 (pps)
*** MaxVelA :** 移動最高速度の配列 (pps)
*** TaccA:** 加速時間の配列 (sec)
*** TdecA:** 減速時間の配列 (sec)
*** SVaccA:** 加速時間のＳ字領域の配列 (pps)
*** SVdecA:** 減速時間のＳ字領域の配列 (pps)
FirstAxisNo : 軸配列の最初の要素
Sync_stop_on : CSTOP 信号が ON したら軸を停止
Cstop_output_on : 軸が異常停止したとき CSTOP 信号を出力する (ALM,EL など)
Sync_option1 : ｺﾏﾝﾄﾞの開始ﾀｲﾐﾝｸﾞの設定
0：初期値（すぐに開始）
1：_7443_start_move_all()ｺﾏﾝﾄﾞか CSTA 信号で開始
2：予約
3：Sync_option2 の条件で開始
Sync_option2 : 例
0：初期値（無意味）
1：軸０の停止で開始
2：軸１の停止で開始
4：軸２の停止で開始
8：軸３の停止で開始
5：軸０と軸２の停止で開始
15：軸０～軸３の停止で開始
stop_mode : 停止ﾓｰﾄﾞの設定
0：急停止
1：減速停止

@ 復帰ｺｰﾄﾞ

ERR_NoError	: 0
ERR_SpeedError	: 11

6.19　汎用ＴＴＬ出力

@ 名前

_7443_d_output	***ﾃﾞｼﾞﾀﾙ出力信号制御***
_7443_get_dio_status	***ﾃﾞｼﾞﾀﾙ出力の状態の読み込み***

@ 記述

_7443_d_output()

汎用ＴＴＬﾃﾞｼﾞﾀﾙ出力ﾋﾟﾝの ON/OFF 状態を制御します。

_7443_get_dio_status()

汎用ＴＴＬﾃﾞｼﾞﾀﾙ出力ﾋﾟﾝの状態を読み込みます。

@ 構文

C/C++ (DOS, Windows 95/NT)

```
I16 _7443_d_output(I16 CardNo, I16 Ch_No, I16 value);
I16 _7443_get_dio_status(I16 CardNo, U16 *dio_sts);
```

Visual Basic (Windows 95/NT)

```
B_7443_d_output (ByVal CardNo As Integer, ByVal Ch_No As Integer, ByVal
                 value As Integer) As Integer
B_7443_get_dio_status (ByVal CardNo As Integer, dio_sts As Integer) As
                 Integer
```

@ パラメータ

CardNo:	基板番号
Ch_No:	ﾁｬﾝﾈﾙ番号 (0 ～ 5)
Value:	出力の ON/OFF 値 Value=0：　出力 OFF Value=1：　出力 ON
dio_status:	出力ﾋﾟﾝの状態 ﾁｬﾝﾈﾙ 0～5 それぞれに対応するﾋﾞｯﾄ 0～5

@ 復帰ｺｰﾄﾞ

ERR_NoError	: 0
ERR_DioNoError	: 27

7

接続例

この章では PPCI7443 とｻｰﾎﾞﾄﾞﾗｲﾊﾞ、ｽﾃｯﾋﾟﾝｸﾞﾓｰﾀﾄﾞﾗｲﾊﾞとの接続例を紹介します。

7.1 配線の概略

CN1: 外部電源(DC24V)入力ｺﾈｸﾀ
CN2: PPCI7443 とﾊﾟﾙｽ列ｻｰﾎﾞﾄﾞﾗｲﾊﾞ、ｽﾃｯﾋﾟﾝｸﾞﾓｰﾀﾄﾞﾗｲﾊﾞとを接続するﾒｲﾝｺﾈｸﾀ
CN3: 手動ﾊﾟﾙｻｰからのﾊﾟﾙｽ指令入力ｺﾈｸﾀ
CN4: 複数の基板の同時ｽﾀｰﾄ/ｽﾄｯﾌﾟを行うｺﾈｸﾀ
CN5: 汎用ﾃﾞｼﾞﾀﾙ出力ｺﾈｸﾀ

図 7.1 は PPCI7443 を使用したｼｽﾃﾑを統合する方法を示します。

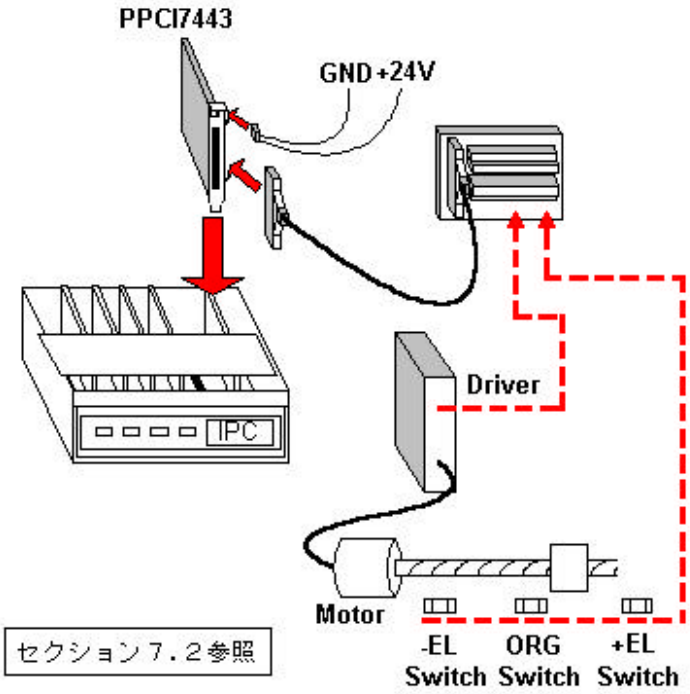


図 7.1 PPCI7443 によるｼｽﾃﾑ統合

7.2 ｻｰﾎﾞﾄﾞﾗｲﾊﾞとの接続例

このｾｸｼｮﾝで PPCI7443 の接続方法の説明としてﾊﾟﾅｿﾆｯｸｻｰﾎﾞﾄﾞﾗｲﾊﾞを使用した例を紹介します。図 7.2 は配線接続です。

注意事項:

1. 便宜上、図面は１軸だけの接続を示しています
2. ﾊﾟﾙｽ出力ﾓｰﾄﾞは OUT/DIR ﾓｰﾄﾞです
   ﾊﾟﾙｽ入力ﾓｰﾄﾞは４逓倍の AB 相ﾓｰﾄﾞです
   ﾕｰｻﾞｰはｿﾌﾄｳｪｱ機能で他のﾓｰﾄﾞに設定することができます
3. 殆どのｻｰﾎﾞﾄﾞﾗｲﾊﾞはﾄﾙｸﾓｰﾄﾞ、速度ﾓｰﾄﾞ、位置ﾓｰﾄﾞで運転します。
   PPCI7443 と接続するにあたって、ﾕｰｻﾞｰは操作ﾓｰﾄﾞを位置ﾓｰﾄﾞに設定します。

Wiring of PPCI7443 with Panasonic MSD

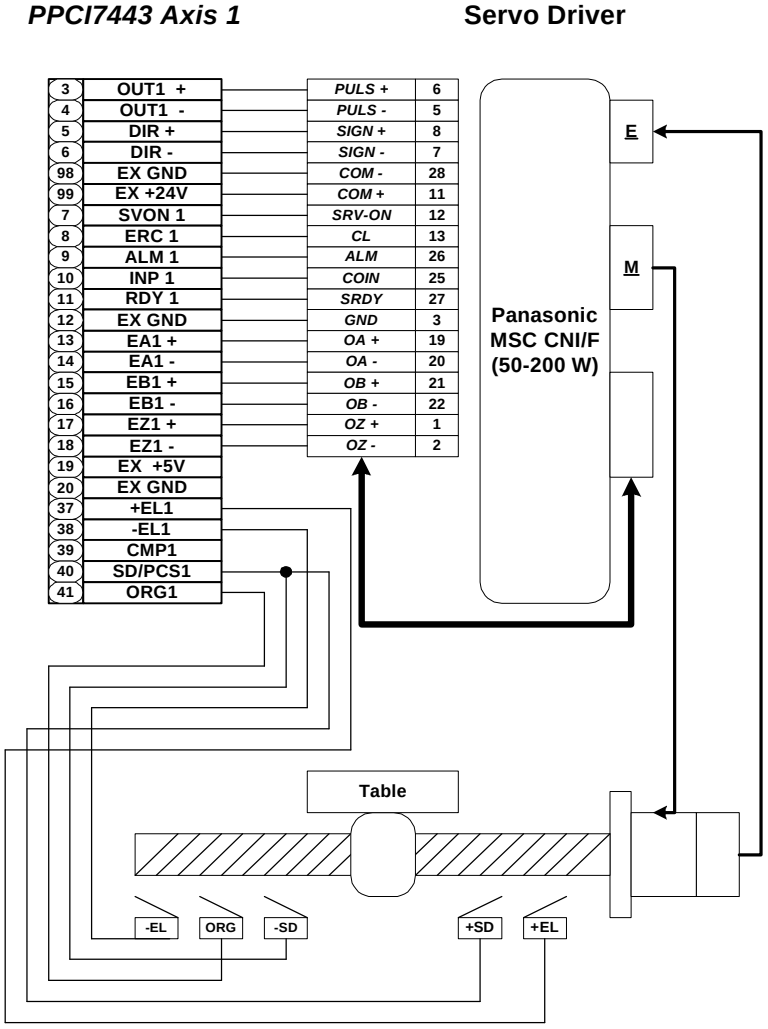


図 7.2 ﾊﾟﾅｿﾆｯｸｻｰﾎﾞﾄﾞﾗｲﾊﾞと PPCI7443 の接続

Wiring of PPCI7443 with SANYO AC Servo PY2

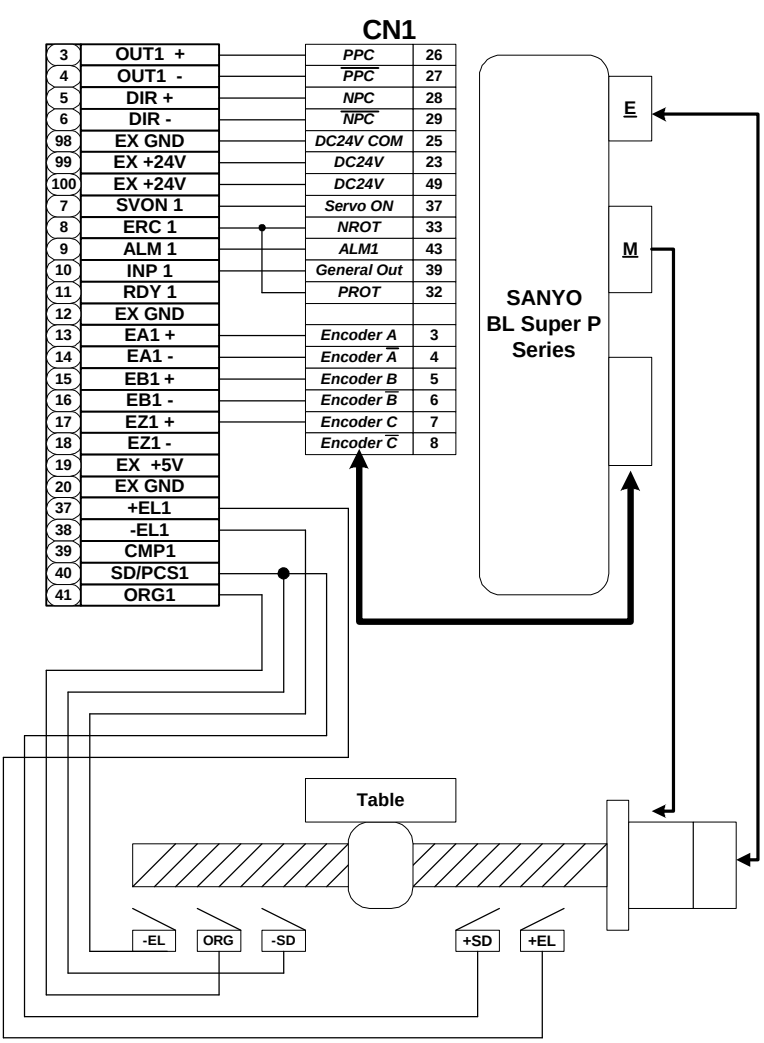


図 7.3 山洋ﾄﾞﾗｲﾊﾞと PPCI7443 の接続